Universidad Nacional Autónoma de México
Escuela Nacional de Estudios Profesionales
Campus Aragón

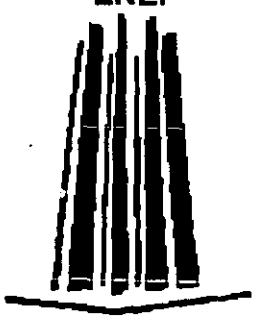
Administración de Base de Datos Oracle
a Fin de Garantizar Seguridad y Mejorar
Tiempos de Respuesta

Tesis que para obtener el Título de
Ingeniero en Computación
Presenta:

Francisco Javier Flores Arellanes

Director de Tesis:
Ing. Juan Gastaldi Pérez

ENEP



ARAGON

México D.F
Agosto del 2001

Prefacio

Hoy no es la conclusión
de mi carrera, esta es pues
un compromiso de vida
que nos reclama vanguardia
y experiencia.

Hoy puedo decir con orgullo
que soy Ingeniero de la UNAM,
pero el conocimiento avanza
y nosotros debemos marchar
a su paso.

Al estudio consciente...
...fuente del saber.

INDICE

pp

INTRODUCCION

Comenzamos con una frase quizá muy trillada pero bastante relevante en nuestro tiempo: "vivimos en la era de la información" y ante tal circunstancia las grandes empresas han emprendido una guerra sin tregua ante la incomunicación, así, en los últimos años, la tecnología de información ha evolucionado de ser un componente básico del éxito del negocio, hasta ser un elemento crítico, requerido para la ejecución de las estrategias del negocio.

El diseño de las políticas de comunicación entonces se ha visto bifurcado y el nuevo reto que enfrentan hoy día las empresas es impactante: deben responder y reaccionar a las crecientes demandas y variantes de la información; solamente empresas que realmente deseen permanecer en pie sufren en este momento o han sufrido la necesidad de nuevos sistemas de almacenamiento que satisfagan ampliamente sus constantes necesidades, siendo la principal necesidad el disponer a cada momento de la información pertinente requerida y así poder tomar decisiones a un plazo muy corto que finalmente determinará el éxito y la propia supervivencia.

De aquí surgen algunas cuestiones como cuál debe ser el sistema de almacenamiento que deben adoptar, qué ventajas debe tener sobre otros que actualmente se encuentran en el mercado y que tan eficiente y eficaz va a resultar en la empresa y los costos que va a representar.

En la actualidad sabemos que un servidor de bases de datos debe ser la llave para la solución de problemas de manejo de información. En general, un servidor debe manejar una gran cantidad de datos en un entorno multiusuario tal que múltiples usuarios puedan accesar a los mismos datos al mismo tiempo. Todo esto debe ser completado con una ejecución muy eficiente. Un servidor de bases de datos debe también prever autorización de acceso y soluciones eficientes para recuperación en caso de fallas.

Lógicamente un sistema requiere del factor humano, una persona o un conjunto de personas que se dediquen exclusivamente a la tarea de dar mantenimiento al sistema de almacenamiento (Administrador de base de Datos DBA), no estoy hablando aquí de sistemas de información, sino de los medios físicos y lógicos que permitan que este conjunto de problemas siempre tengan una solución, de esto dependerá que los sistemas de información a los que esté dando servicio este medio de almacenamiento tengan el impacto deseado.

Entonces, tenemos como alternativa al oracle server, el cual provee una solución eficiente para este problema y es precisamente de este sistema de almacenamiento que voy a tratar en esta investigación, iniciando con la arquitectura del oracle server, abarcando temas acerca de la seguridad de la información, el respaldo y la recuperación de los datos y finalmente sobre la afinación o puesta a punto de la base de datos a fin de garantizar un rendimiento óptimo en todos los procesos que hagan uso del sistema de almacenamiento oracle.

Un servidor de base de datos Oracle ofrece una gran facilidad de uso y alta potencia. Está configurado para entornos actuales de grupos de trabajo dinámicos e incluye un conjunto de herramientas de gestión fáciles de usar. En el entorno informático en constante cambio, de hoy en día, los usuarios demandan un acceso rápido y eficaz a la información. Los empresas exigen soluciones a un bajo costo que le permitan tomar decisiones rápidamente ante los cambios constantes en sus negocios y exigen que estas soluciones sean fáciles de manejar, potentes y económicas. Oracle es una solución de sistemas que satisface las demandas de las empresas por que esta basado en una arquitectura avanzada y escalable, Oracle es un servidor de base de datos universal, su uso reduce los costos y al mismo tiempo aprovecha las funcionalidades proporcionadas por las plataformas de hardware sobre las que está disponible, abarcando desde entornos con un solo procesador, hasta entornos multiprocesador. Además incluye funciones y servicios necesarios para construir aplicaciones críticas para el negocio, asegurando un acceso eficiente y fiable a los datos y una gestión de una forma fácil y completa.

Luego de un análisis y la toma de un conjunto de problemas que se suscitan día a día en el medio laboral con respecto a la difícil tarea de la administración de los distintos sistemas de almacenamiento que se manejan, pienso es de alta importancia dar inicio a una investigación exhaustiva con respecto a la administración de la base de datos oracle (Oracle Server), en primera instancia por que aun en nuestro país son pocas las personas que se dedican a tan importante tarea y en segunda instancia por que esas pocas personas son un grupo demasiado cerrado debido a lo complejo de sus actividades. Una investigación de esta magnitud traería consigo primero explicar de una manera bastante clara sobre las principales tareas del Administración del Oracle Server y al mismo tiempo proponer como una solución continua, una serie de procesos que permitan dar mantenimiento a las actividades realizadas por el DBA.

El resultado de esta investigación beneficiaría directamente a cualquier persona que se dedique a la administración de la base de datos Oracle pues aquí encontraría un conjunto de recursos que estarían a su disposición en la solución de sus problemas, principalmente de tiempos de respuesta que sería el principal enfoque de la investigación, así como también se verían beneficiadas las empresas que actualmente usen como medio de almacenamiento al Oracle server, pues trataría de darse aquí una solución genérica para los problemas más comunes en cuestión de la administración del Oracle server, y específicamente la inquietud de desarrollar este tema es a raíz de los problemas a los que un grupo de personas que laboramos en la empresa Fianzas Monterrey hemos tenido que enfrentar tales como la ejecución de las tareas de la base de datos oracle que allí se manejan, resultando en la mayoría de las veces en una crisis caótica debido a la falta de experiencia de los actuales administradores en este ramo y más aun, el acceso a la información de este rubro es un tanto limitado debido a que existe poca bibliografía y la mayoría de las veces esta se encuentra en el idioma inglés que hace poco más complejo su entendimiento.

Todas las tareas de administración son de aplicación diaria, estas van desde el compilar un proceso en un ambiente de producción, hacer migración y carga de datos y ejecutar procesos, hasta monitorear constantemente la carga de trabajo del sistema para generar estadísticas y ver de esta manera cuales son las acciones a realizar bajo circunstancias dadas, sin embargo se deben conocer a fondo los conceptos relacionados a las áreas en las que se va a estar trabajando así como las estructuras y métodos para hacer las actividades propias.

En primera instancia, se debe entender a la perfección precisamente sobre el oracle server, este es un sistema de administración para bases de datos relacionales, y que tiene un alto grado de aprovechamiento para el manejo de la información. Un oracle server está dividido en una base de datos y una instancia. Una base de datos oracle tiene una estructura física y una lógica. Debido a que la parte física y lógica están separadas en la estructura del servidor, la parte física almacena los datos sin que afecte al acceso a la estructura lógica. La estructura física de la base de datos oracle está determinada por el sistema operativo, a través de un conjunto de archivos que constituyen la base de datos. La estructura lógica está determinada por tablespaces, segments y extents, estos tienen como fin la organización del espacio físico que usará la base de datos, en el primer capítulo de esta investigación se trata muy a detalle como esta compuesta cada una de las partes de la base de datos y el conjunto de tareas que se tienen que llevar a cabo para su mantenimiento.

Con respecto a la seguridad de la base de datos que se maneja en el segundo capítulo, podemos decir en forma general que es la manera en que se deben asignar permisos a los usuarios para ejecutar un conjunto de acciones sobre la base de datos y los objetos que pertenecen a esta, tomando en consideración los privilegios que estos tengan. Este control como se menciona, regula el acceso a todos los usuarios hacia los objetos de la base de datos a través de privilegios. Un privilegio es un permiso para accesar a un objeto dado de una manera preestablecida, por ejemplo, permisos para poder consultar una tabla, borrarla, hacer consultas, etc., en este segundo capítulo se enumeran cada uno de los niveles de seguridad con que cuenta el oracle server así como niveles adjuntos propios de los sistemas operativos y las plataformas de red.

Una de las tareas más importantes es la que se desarrolla en el tercer capítulo y es precisamente la de respaldo y recuperación de la base de datos. El administrador de la base de datos debe estar preparado para dar solución a posibles fallas en el sistema ya sean de hardware, software, red o procesos. Si estas fallas afectan la operación de la base de datos, se debe recuperar y retornar a la operación normal lo más rápidamente posible. La recuperación debería proteger la base de datos y los usuarios asociados para evitar problemas innecesarios y reducir las posibilidades de tener que duplicar el trabajo manualmente. Un proceso de recuperación varía dependiendo del tipo de falla que ha ocurrido, las estructuras que afectó y el tipo de recuperación que se ejecutará. Si se perdieron o se dañaron algunos archivos, la recuperación puede aumentar para no reiniciar una instancia, si los

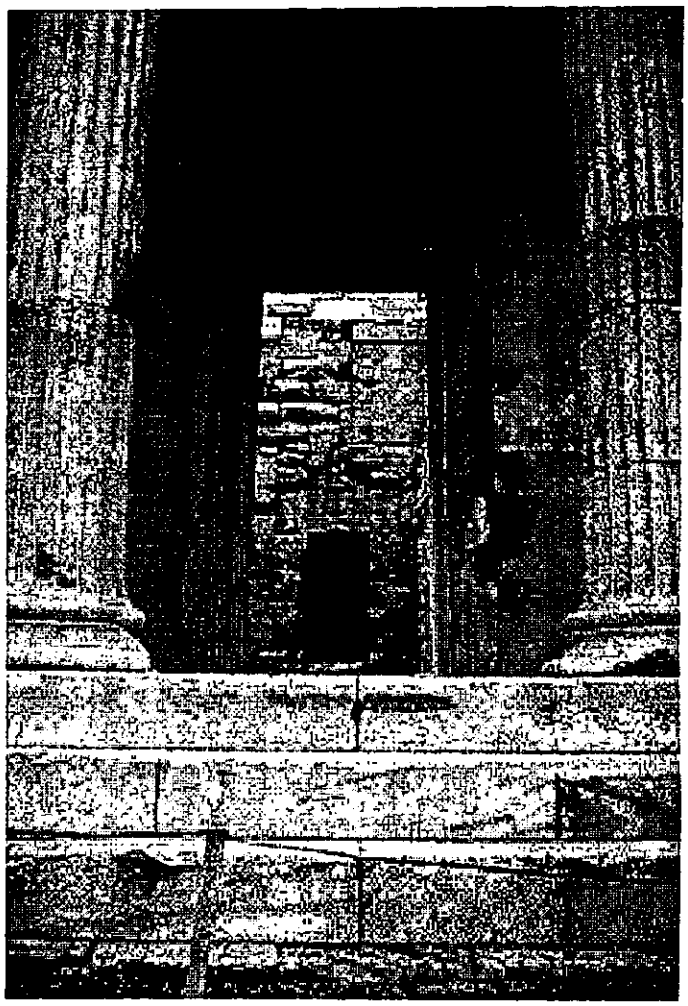
datos se han perdido, la recuperación requerirá pasos adicionales, aquí se mencionan cuales son las estructuras más importantes que se deben respaldar así como las técnicas para realizar todas estas tareas importantísimas.

Finalmente y sumándose a las demás actividades de la administración y como capítulo final de esta investigación es todo lo relacionado al rendimiento y afinación, partiendo del hecho que el principal problema que tiene casi cualquier base de datos (Incluyendo la de fianzas Monterrey) es su constante cambio. Los cambios pueden darse de distintas formas, por ejemplo, las tablas de la base de datos crecen en cantidad de registros, caso especifico es cuando una tabla de la base de datos en su operación normal espera un crecimiento del 10% mensual y de pronto existe una liberación de alguno de los sistemas que usan esa tabla en alguna de las dependencias de la empresa, entonces se realiza una carga de datos con archivos históricos que generan un crecimiento de hasta un 30 o 40% no esperado. Las instancias crecen en número y las aplicaciones desde un inicio están mal distribuidas. Estos cambios son la a causa para entrar en un modo de crisis sobre el rendimiento y los tiempos de respuesta de los sistemas de información a los que da servicio el oracle server, por ejemplo el sistema Confianza2000 y Confianza Web que día a día crece por su cantidad de usuarios y por la cantidad de información que se procesa a través de estos.

Debido a lo anterior, se invierte demasiado tiempo entre la administración del Oracle server, la modificación de los objetos para soportar situaciones no previstas, en el diseño y la afinación de los procesos para mejorar el rendimiento, o bien el administrador gasta muchas noches y fines de semana en modificar y supervisar el rendimiento del sistema. Lo ideal aquí es que con esta investigación se lleve a una planeación más adecuada, donde el esfuerzo para realizar la administración y afinación de la base de datos Oracle sea mínimo y por consiguiente nulificar o poner en un rango muy bajo el nivel de crisis que hasta el momento está en constante repunte. Aquí al igual que en los demás capítulos se mencionan puntos muy específicos en los cuales hay que poner atención a fin de mantener óptimo en rendimiento del sistema.

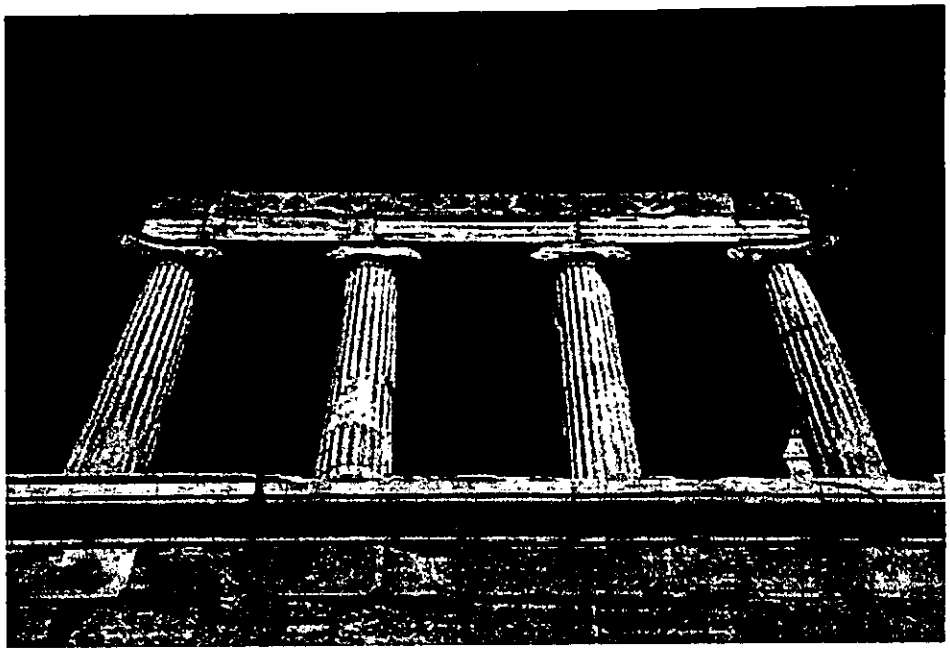
Para la elaboración de este trabajo de investigación tuve a bien el encontrar con algunas referencias bibliográficas que explican muy a detalle cada uno de los temas que yo manejo aquí, hay bastante bibliografía incluso muy explícita en cuanto a su contenido, es decir, por un lado contaba con libros y manuales que solo hablan de la arquitectura del oracle server y sobre tareas de administración y por otro, libros que solamente desarrollan los temas de afinación del rendimiento por ejemplo. Caso difícil, no existe bibliografía en español, absolutamente toda la bibliografía fue consultada en el idioma inglés, incluyendo documentación extraída de la internet de la página de oracle. El contenido de los capítulos trata de resolver de la manera más cercana el problema de el qué, por que más que un manual de referencia, se plantea este trabajo como una guía que nos brinda un panorama amplio de cómo interpretar y dar solución a cada uno de los problemas que aquí se manejan.

ADMINISTRACION DE LA BASE DE DATOS ORACLE A FINDE GARANTIZAR SEGURIDAD Y MEJORAR TIEMPO DE RESPUESTA



Capítulo I

Arquitectura del Oracle Server



EL ADMINISTRADOR DE BASES DE DATOS (DBA) Y LA BASE DE DATOS

Antes de comenzar con la arquitectura del oracle server es conveniente saber que el responsable de las tareas de administrar una base de datos es una persona que basada en sus conocimientos de oracle tendrá como principal responsabilidad la realización de las siguientes actividades:

- Instalar y actualizar el oracle server y las herramientas de aplicación
- Crear una base de datos primaria, estructuras de almacenamiento y objetos primarios.
- Localizar el sistema de almacenamiento y planear futuros requerimientos de sistema de base de datos.
- Modificar la estructura de la base de datos.
- Enrolar usuarios.
- Controlar y monitorear el acceso de los usuarios a la base de datos.
- Hacer respaldos y recuperaciones de la base de datos.
- Mantener la seguridad del sistema.
- Monitorear y optimizar el rendimiento de la base de datos (tiempo de ejecución y respuesta).

Esta persona la vamos a denominar como Administrador de Base de Datos (DBA). Además el DBA debe entender muy bien sobre los componentes tales como procesos, estructuras de memoria y archivos.

Partimos también de que oracle es un sistema cliente-servidor, esto quiere decir que el sistema permite partir los procesos entre el servidor de la base de datos y las aplicaciones a nivel cliente[1]. La computadora corre o ejecuta el sistema de administración de la base de datos el cual controla todas las responsabilidades del servidor mientras que la estación de trabajo(cliente) corre la aplicación concentrándose en la interpretación y consulta de datos.

El oracle server es un sistema de administración para una base de datos relacional, es abierto, comprensible y tiene un alto aprovechamiento para el manejo de la información. Un oracle server consiste en una base de datos y una instancia.

Una base de datos oracle tiene una estructura fisica y una lógica. Debido a que la parte física y lógica están separadas en la estructura del servidor, la parte física almacena los datos sin que afecte al acceso a la estructura lógica. La estructura física de la base de datos oracle está determinada por el sistema operativo, a través de un conjunto de archivos que constituyen la base de datos. La estructura lógica está determinada por tablespaces, segments y extents[2], estos dictan cómo el espacio físico de la base de datos será usado.

Una instancia oracle es la estructura de memoria: el System Global Area (SGA) está en memoria y un conjunto de procesos Background son iniciados también. La SGA es un área en memoria usada por información de la base de datos y es un área compartida por los usuarios de la base de datos. La combinación de los Procesos Background y los buffers de memoria son llamados una instancia .

Una instancia tiene dos tipos de procesos, los procesos de usuario y los procesos de oracle. Un proceso de usuario ejecuta código desde una aplicación (por ejemplo oracle Forms) o una herramienta de oracle (tal como el server manager). Un proceso de oracle es un proceso en el servidor que realiza un trabajo para que un proceso de usuario pueda funcionar en el oracle server.
Tenemos pues el preámbulo inicial que nos define los elementos que van a tomar parte en el desarrollo de la investigación, estos son el DBA, el sistema cliente-servidor, la base da datos y la instancia, dando inicio definiendo estas estructuras de memoria y procesos.

---

[1] En el entorno de computación actual, un ordenador Macintosh, Windows, UNIX o una computadora grande, puede ser un cliente Cualquiera de estas plataformas puede actuar como servidor e incluso puede actuar como cliente y servidor simultáneamente. Esta doble función es posible debido a las capacidades multitarea de los modernos sistemas operativos.

[2] Una observación necesaria es que la gran mayoría de los términos empleados para referirse a los objetos que componen la base de datos a pesar de tener traducción, es común que se usen en su idioma original que es el inglés, incluyendo a la misma empresa de oracle donde se imparten cursos y se usan coloquialmente en este idioma.

INSTANCIA ORACLE

Siempre una base de datos es levantada sobre un servidor de bases de datos, oracle está asignado en un área de memoria denominada System Global Area (SGA) e inicia uno o más procesos oracle. La combinación de el SGA y los procesos oracle es llamada una Instancia de base de datos oracle. La memoria y los procesos de una instancia trabajan de manera que el manejo de los datos sea eficiente y sirva a uno o a múltiples usuarios de la base de datos.

Oracle levanta una instancia y la base de datos se monta sobre la instancia. Se pueden levantar múltiples instancias sobre la misma computadora y cada una podrá almacenar y accesar a una base de datos física propia.

Estructura de Procesos

Un proceso es un mecanismo de un sistema operativo que permite ejecutar una serie de pasos. Un proceso normalmente tiene un área de memoria privada sobre la cual corre. La estructura de proceso de oracle es importante por que define como las diferentes actividades deben ser ejecutadas, esto trae como resultado la maximización del rendimiento.

Existen dos tipos de procesos, los procesos simples que constituyen un sistema oracle en el cual el código será ejecutado por un solo proceso; esto quiere decir que solamente un usuario puede accesar a una instancia oracle en un entorno en el cual múltiples usuarios no pueden accesar la base de datos al mismo tiempo, generalmente esto se da con el trabajo en una PC.

Un proceso múltiple en cambio usa diferentes procesos para ejecutar diferentes partes de oracle y separar proceso para cada usuario de oracle que esté conectado, para cada usuario la instancia oracle ejecuta una tarea especifica. A través de la división del trabajo de oracle en diferentes procesos, múltiples usuarios y aplicaciones pueden estar simultáneamente conectados a una instancia y el sistema mantiene un excelente rendimiento.

Cada usuario que está conectado tienen procesos separados y diferentes procesos background que son usados para ejecutar oracle. En la **figura 1** se puede observar a múltiples usuarios corriendo una aplicación sobre la misma máquina, pero en este caso la computadora es una mainframe o minicomputadora.

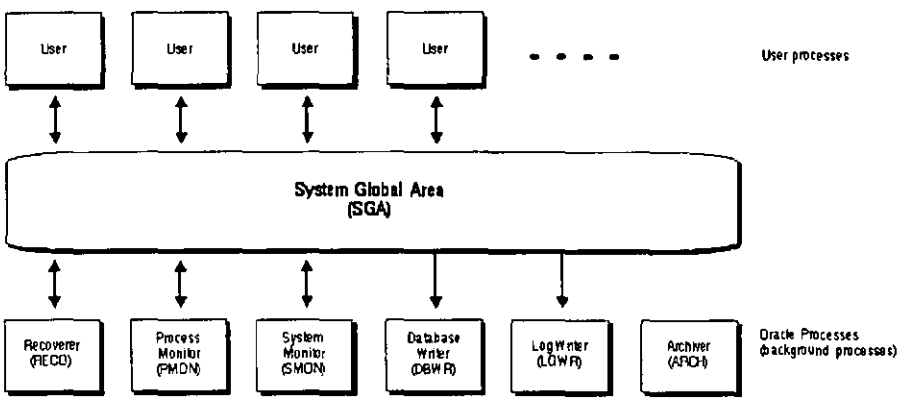


Fig. 1

Cuando un usuario corre un programa de aplicación como Pro *C, o alguna herramienta de oracle como el server manager, oracle crea un proceso de usuario para correr esa aplicación de usuario.

En un sistema de procesos múltiples existen dos diferentes tipos de procesos: los procesos de servidor y los procesos background. Oracle crea un proceso de servidor o server processes para manejar los requerimientos de los proceso de usuarios conectados a la instancia. Comúnmente cuando una aplicación y oracle operan en la misma máquina en lugar de trabajar sobre una red, el user process y su correspondiente server process son combinados en un solo proceso para reducir el uso del sistema. Sin embargo, cuando una aplicación y oracle operan sobre diferentes máquinas, el user process se comunica con oracle via un server process

separado. Para maximizar el rendimiento y dar servicio a múltiples usuarios, un proceso múltiple de oracle usa algunos procesos adicionales llamados *procesos background*

Una instancia de Oracle puede tener muchos procesos background, que no necesariamente están presentes. Entre los procesos background que están en una instancia oracle podemos encontrar los siguientes:

- Database Writer (DBWR)
- Log Writer (LGWR)
- Checkpoint (CKPT)
- System Monitor (SMON)
- Process Monitor (PMON)
- Archiver (ARCH)
- Recover(RECO)
- Dispatcher (Dnnn)
- Server (Snnn)

En la **figura 2**[3] podemos observar una instancia Oracle, cada uno de los proceso background así como su interacción con las diferentes partes de la base de datos oracle

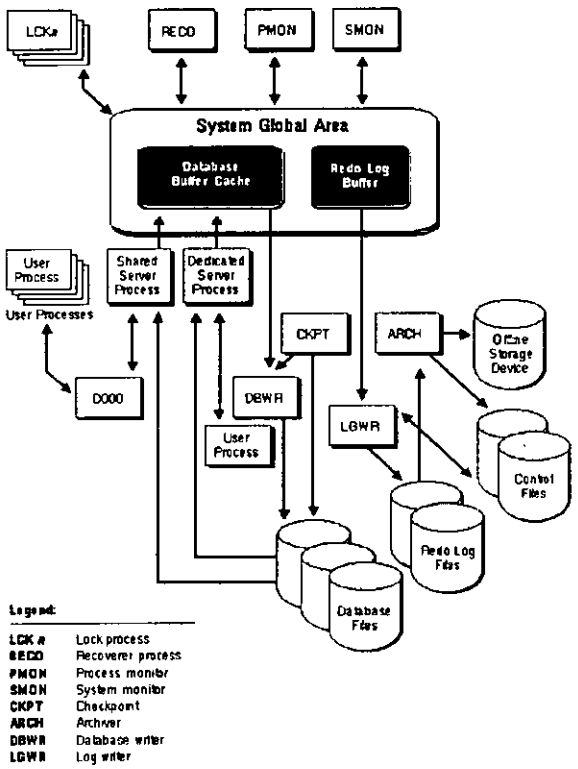


Fig. 2 Instancia Oracle

Database Writer (DBWR)

Este proceso escribe los buffers a los datafiles. DBWR es un proceso background de oracle que es responsable del manejo del buffer caché. Cuando un buffer del buffer caché es modificado, este es marcado. El primer trabajo del DBWR es guardar el buffer caché escribiendo la información que contenga en el disco. Un algoritmo LRU (el más recientemente utilizado) guarda los datos más recientemente usados en memoria y de esta forma se minimiza la I/O. El DBWR guarda bloques que son usados frecuentemente, por ejemplo, bloques de tablas o índices pequeños que constantemente son accesados de tal forma que estos

[3] BOBROWSKI, Steven, Oracle 7 Server Concepts. pp 122

bloques no sean leídos de nueva cuenta desde el disco. Este mismo proceso, remueve los bloques que son poco accesados (como bloques que son parte de tablas o índices muy grandes) desde el SGA.

Log Writer (LGWR)

El proceso LGWR escribe la información que se encuentra en los redo log buffers en un redo log file sobre el disco; es el responsable del manejo del redo log buffer. LGWR escribe todas las entradas redo que han sido copiadas en el buffer desde la última vez que se escribió sobre el disco. LGWR escribe una porción del buffer en el disco y tiene como tareas:

- Guarda los registros una vez que un proceso de usuario haga commit a una transacción.
- Reconstruye los buffers cada tres segundos
- Reconstruye los buffers cuando un redo log buffer está lleno
- Reconstruye los buffers cuando el DBWR escribe sobre el disco los buffers que fueron modificados.

LGWR escribe sincrónicamente sobre los archivos del grupo en forma de espejo. Si uno de los archivos se daña o no esta disponible, LGWR puede continuar escribiendo sobre los otros archivos del grupo. Si todos los archivos en un grupo se dañan o el grupo no esta disponible por que este no ha sido archivado, LGWR no puede continuar con su función.

El redo log buffer es un buffer circular, cuando el LGWR escribe las entradas redo desde el redo log buffer, el server processes puede copiar sobre la entradas de el redo log buffer que ya han sido escritas en el disco. LGWR generalmente escribe bastante rápido para asegurar que el espacio esté siempre disponible en el buffer para nuevas entradas. Si el proceso checkpoint no está presente, LGWR es también el responsable de hacer un registro de checkpoints.

Checkpoint (CKPT)

Cuando un checkpoint ocurre, oracle debe actualizar las cabeceras de los data files para indicar el checkpoint. En situaciones normales este trabajo es ejecutado por LGWR, sin embargo si los checkpoint no se presentan correctamente sobre la ejecución del sistema (generalmente cuando existen muchos datafiles), se puede habilitar el proceso CKPT para separar el trabajo entre el CKPT y LGWR. Para muchas aplicaciones el CKPT no es necesario. Si una base de datos tiene muchos datafiles y el rendimiento de LGWR se reduce significativamente durante los checkpoints es necesario habilitar el CKPT.

Este proceso no escribe bloques en el disco; DBWR ejecuta siempre esta tarea. Las estadísticas de checkpoints DBWR mostradas por el System_Statistics monitoreadas sobre el server manager muestran el número de mensajes checkpoints que han sido completados, siendo indiferente si el proceso CKPT esta habilitado o no.

System Monitor (SMON)

El proceso SMON ejecuta una recuperación de la instancia cuando esta se inicia. SMON es también responsable de limpiar los segmentos temporales que no se han usado últimamente, esto trae consigo que un espacio importante de extents estén disponibles.

Process Monitor (PMON)

El PMON ejecuta una recuperación cuando un proceso de usuario falla; es el responsable de limpiar el caché y liberar los recursos que un proceso ha usado. Por ejemplo, cuando se hace un reset a una transacción activa sobre una tabla, remueve el process ID en la lista de procesos activos. PMON también checa periódicamente el estatus del dispatcher y server processes, y reinicia algunos procesos muertos siempre y cuando no hayan sido matados intencionalmente.

Recover (RECO)

El proceso RECO es usado con la opción distribuida que automáticamente resuelve fallas envueltas en transacciones distribuidas. El RECO de un nodo automáticamente se conecta a otras bases de datos envueltas en una transacción que está en duda. Cuando el RECO restablece la conexión con la base de datos, automáticamente se resuelve la transacción "en duda".

El RECO remueve automáticamente los registros correspondientes a alguna transacción dudosa, asi como intenta establecer comunicación con un servidor remoto que no ha sido restablecido, RECO automáticamente intenta conectarse nuevamente después de un intervalo de tiempo. Este proceso solo está presente si el sistema permite transacciones distribuidas.

Archiver (ARCH)

El proceso ARCH copia los redo log files online en un medio de almacenamiento designado una vez que este se llena. ARCH está presente solamente cuando el redo log es usado en modo ARCHIVELOG y automáticamente se habilita la archivación.

Dispatcher Processes (Dnnn)

El proceso dipatcher permite a los procesos de usuario compartir un número limitado de procesos de servidor. Sin un dispatcher, cada proceso de usuario requeriría un proceso de servidor dedicado, empero, con el servidor multitarea menos procesos de servidor son requeridos por el mismo número de usuarios. De esta manera, en un sistema con muchos usuarios, el servidor multitarea puede soportar un gran número de usuarios particularmente en un entorno cliente-servidor dónde la aplicación del cliente y el servidor operan en diferentes máquinas.

Se pueden crear múltiples proceso dispatcher para una sola instancia; para cada dispatcher debe ser creado un protocolo de red usado por oracle. El DBA tendría que iniciar un número óptimo de procesos dispatcher dependiendo de las limitaciones del sistema operativo y agregar o borrar dipatchers mientras la instancia esté corriendo.

Cuando una instancia es iniciada, el listener abre y establece una comunicación a través de la cual los usuarios se conectan a oracle. Entonces, cada dispatcher da al listener una dirección en la cual el dispatcher escucha para conexiones requeridas. Cuando un proceso de usuario hace un requerimiento de conexión el listener examina el requerimiento y determina si el usuario puede usar un dispatcher. Si es asi, el listener retorna la dirección de el dispatcher y a su vez cargado directamente al dispatcher.

ESTRUCTURAS DE MEMORIA DE ORACLE

Oracle usa la memoria para almacenar la siguiente información:

- Código de programas que ha comenzado a ejecutarse
- Información acerca de la sesión conectada, siempre que esta no esté actualmente activa
- Datos necesarios durante la ejecución de un programa
- Información que es compartida entre los procesos de oracle
- Información oculta que esta permanentemente almacenada en memoria.

Las estructuras de memoria básicas asociadas con oracle son:

- Software code áreas
- El system global area (SGA)
- Program global areas
- Sort areas

Memoria Virtual

En muchos sistemas operativos, oracle toma ventaja de la memoria virtual. La memoria virtual es una caracteristica del sistema operativo que ofrece aparentemente más memoria y es proveída por memoria real haciendo más flexible el uso de la memoria principal. La memoria virtual simula memoria usando una combinación de memoria real y un almacenamiento secundario (generalmente espacio en disco).

Software Code Areas

Esta es una porción de memoria usada para almacenar código que ha comenzado a ejecutarse. El código de oracle es almacenado en la software area la cual es comúnmente una localidad diferente para los programas de usuario como una más exclusiva o localidad protegida.

Las software areas generalmente son estáticas en tamaño, solamente cambian cuando el software es actualizado o reinstalado, el tamaño de las áreas varia dependiendo del sistema operativo. Estas áreas son solo de lectura y al ser instaladas pueden estar compartidas o no compartidas. El código de oracle es compartido de tal forma que todos los usuarios de oracle puedan accesar sin tener múltiples copias en memoria, esto resulta en una mejora significativa en el rendimiento de la memoria.

System Global Area (SGA)

La SGA es un grupo de estructuras de memoria compartidas que contienen datos e información de control para una instancia de una base de datos oracle. Si múltiples usuarios están conectados concurrentemente a la misma instancia, los datos en el SGA son compartidos entre los usuarios.

Un SGA y los procesos de oracle constituyen una instancia. Oracle automáticamente asigna la memoria para un SGA cuando se inicia la instancia y la memoria es salvada cuando se da de baja la instancia. Cada instancia tiene su propia SGA.

El SGA es un área de memoria compartida, todos los usuarios conectados a los múltiples procesos de la instancia de la base de datos pueden usar información contenida dentro del SGA, la SGA es también modificable, diferentes procesos la modifican durante la ejecución de oracle.

El SGA tiene las siguientes subdivisiones:

- El database buffer caché
- El redo log buffer
- La shared pool
- El data dictionary caché

El Database Buffer Caché

Está es una porción del SGA que mantiene copias de los bloques de datos leídos desde los datafiles. Todos los procesos concurrentemente conectados a la instancia comparten el acceso al database buffer caché.

Los buffers en el caché están organizados en dos listas: la lista sucia y la lista usada menos recientemente(LRU). La lista sucia mantiene los buffers sucios. Un buffer sucio es un buffer que ha sido modificado pero que todavía no ha sido escrito en el disco. La LRU mantiene los buffers libres, los fijos y los que sucios que aun no han sido movidos de la lista sucia. Los buffers libres son buffers que aun no han sido modificados y están disponibles para su uso, los buffers fijos son buffers que actualmente están siendo accesados.

Cuando un proceso necesita accesar a un bloque que aun no se encuentra en el buffer caché, el proceso debe leer el bloque desde un datafile en el disco y ponerlo dentro de un buffer cache. Antes de hacer la lectura para el caché, el proceso primero debe encontrar un buffer libre. El proceso busca en la LRU hasta que encuentra un buffer libre o hasta que llega al límite máximo de buffers.

La inicialización del parámetro DB_BLOCK_BUFFERS especifica el número de buffers en el database buffer caché. Cada buffer en el caché debe tener el tamaño de un data block de oracle. Sin embargo, cada database buffer caché en el caché puede mantener un solo bloque de datos leído desde un datafile.

Lo primero que debe hacer un proceso de usuario de oracle es extraer una parte de los datos, el proceso debe copiar los datos desde el disco hacia el caché antes de accesar a estos (caché miss). Cuando un proceso accesa una parte de los datos que están

ya sobre el caché puede entonces leer los datos directamente de la memoria(caché hit). Accesar a los datos a través del caché hit es más rápido que accesarlos a través del caché miss.

El Redo Log Buffer

El redo log buffer es un buffer circular dentro del SGA que mantiene la información de los cambios hechos en la base de datos. Esta información es almacenada en entradas redo. Las entradas redo contienen la información necesaria para reconstruir cambios en la base de datos hechos por cláusulas INSERT, UPDATE, DELETE, CREATE, ALTER O DROP. Las entradas redo son usadas por la base de datos en caso de que sea necesaria una recuperación. Las entradas redo copian a través de un proceso de oracle desde la memoria del usuario al redo log buffer del SGA . Las entradas redo toman continuamente un espacio secuencial en el buffer. El LGWR escribe desde el redo log buffer en un redo log file online en el disco.

La inicialización del parámetro LOG_BUFFER determina el tamaño (en bytes) del redo log buffer. En general un tamaño grande reduce las lecturas desde el disco, particularmente si las transacciones son muy numerosas.

La Shared Pool

La shared pool es un área en el SGA que contiene otras tres áreas: el library caché, el dictionary caché y las estructuras de control. La **figura 3**[4] muestra el contenido de la shared pool.

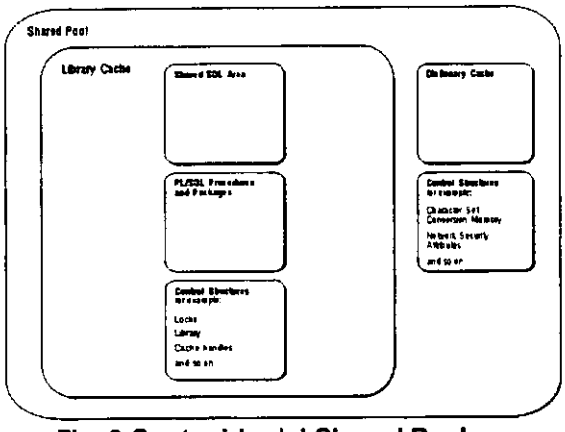


Fig. 3 Contenido del Shared Pool.

El tamaño total de la shared pool está determinado por el parámetro de inicialización SHARED_POOL_SIZE, si se incrementa el valor de este parámetro se incrementa la cantidad de memoria reservada para la shared pool y consecuentemente, este espacio es reservado para las shared SQL áreas.

Library Caché

El library caché contiene shared SQL areas, private SQL áreas, procedimientos y paquetes de PL/SQL y estructuras de control.

Shared SQL Areas y Private SQL Areas

Oracle reconoce cuando dos usuarios están ejecutando el mismo comando SQL y rehusa la misma área de memoria para estos usuarios. Sin embargo, cada usuario debe tener una copia separada del comando en un private SQL área.

Una shared SQL area es un área de memoria que contiene la estructura del parseo y el plan de ejecución de un comando SQL simple. Oracle asigna memoria para el shared pool cuando un comando SQL es parseado y el tamaño de la memoria depende de

[4] BOBROWSKI, Steven, Oracle 7 Server Concepts. pp 130

lo complejo del comando. Una shared SQL área está siempre dentro de la shared pool y es compartida para comandos SQL idénticos.

Una private SQL area, es un área en memoria que contiene datos tales como lazos de información y buffers en tiempo de ejecución. Cada sesión que usa un comando SQL tiene una private SQL area. Cada usuario que ejecuta un SQL idéntico tiene su propia private SQL area que es usada como una shared SQL area sencilla.

El manejo de una private SQL area es responsabilidad de los procesos de usuario,. La asignación y desasignación de una private SQL area depende en gran medida de cual herramienta de aplicación se esté usando, comúnmente el número de private SQL areas que un usuario procesa se puede asignar si siempre se limita a través del parámetro OPEN_CURSORS. El valor de default para este parámetro es 50.

Siempre que las shared SQL areas estén disponibles para múltiples usuarios, el library caché debe estar contenido en la shared pool dentro del SGA. El tamaño de el library caché solo con el data dictionary caché, está limitado por el tamaño de la shared pool. La asignación de memoria para el shared pool está determinada por el parámetro de inicialización SHARED_POOL_SIZE. El valor default para este parámetro es de 3.5 megabytes.

Dictionary Caché

El diccionario de datos es una colección de tablas y vistas que contienen información de la base de datos, sus estructuras y sus usuarios. El tipo de información que se almacena en el diccionario de datos es la siguiente:

- Nombre de las tablas, las vistas y la base de datos.
- Nombres y tipos de datos de las columnas de las tablas
- Privilegios de todos los usuarios de oracle.

La información es usada como material de referencia para los DBA en el diseño de aplicaciones y de igual manera para los usuarios finales. Oracle accesa frecuentemente al diccionario de datos durante el parseo de un comando SQL, este acceso es esencial para la continua operación de oracle.

Debido a que el diccionario de datos es accesado frecuentemente por oracle, existen dos áreas especiales en memoria que mantienen el diccionario de datos. Un área es llamada el data dictionary caché también conocida como el row caché y otra área es conocida como el library caché. El data dictionary caché es compartida por todos los proceso de usuario de oracle.

Tamaño del SGA

El tamaño del SGA está determinado en el levantamiento de la instancia. Para una ejecución óptima en los sistemas, el SGA debería ponerse en memoria real. Si la SGA no es puesta en memoria real y parte de ella se usa en memoria virtual, la ejecución del sistema de base de datos podria venirse abajo dramáticamente.

Los parámetros que más afectan el tamaño del SGA son los siguientes:

DB_BLOCK_SIZE. El tamaño en bytes de un datablock sencillo y database buffer.
DB_BLOCK_BUFFERS. El número de database buffers, el total del parámetro DB_BLOCK_SIZE, asignado por el SGA.
LOG_BUFFER. El número de bytes asignados por el redo log buffer.
SHARED_POOL_SIZE. El tamaño en bytes del área asignada para el shared SQL y comandos PL/SQL.

Parte del SGA contiene información general acerca del estado de la base de datos y la instancia y cuales proceso background necesitan ser accesados esta parte es llamada fixed SGA, no existe datos de usuario almacenados aquí.

Program Global Area (PGA)

El PGA es una región de memoria que contiene datos y control de información para un solo proceso del servidor o procesos background. El PGA es asignado por oracle cuando un proceso de usuario se conecta a la base de datos oracle y es creada una sesión, aunque existan variaciones de configuración del sistema operativo. El contenido del PGA varía dependiendo si la instancia asociada esta corriendo en un servido multitarea.

Configuración de Oracle

Todos los usuarios conectados a oracle deben ejecutar dos módulos de código para accesar a la instancia de la base de datos: la aplicación o herramienta de oracle. Un usuario de la base de datos ejecuta una aplicación o herramienta de oracle (como oracle forms), en la que usa comandos SQL para la base de datos. En una instancia de procesos múltiples el código para los usuarios conectados puede ser configurado en tres diferentes variaciones:

Para cada usuario tanto la aplicación de la base de datos como el oracle server, el código es combinado en un solo proceso de usuario. Para cada usuario, la aplicación de base de datos está corriendo por un proceso diferente y es el primero que ejecuta el código del oracle server. Esta configuración es llamada Arquitectura dedicada de servidor.

La aplicación de la base de datos es un proceso diferente y el primer proceso ejecuta el código del oracle server, además cada server process que ejecuta el código puede a su vez servir a múltiples procesos de usuario. Esta configuración es llamada Arquitectura de servidor multitarea.

Una conexión es una comunicación entre un proceso de usuario y una instancia de oracle. La comunicación se establece usando mecanismos de interprocesos de comunicación.

Una sesión es una conexión específica de un usuario a una instancia de oracle vía un proceso de usuario; por ejemplo, cuando un usuario inicia SQL plus, el usuario debe proveer un nombre de usuario válido y una contraseña para poder establecer la sesión para el usuario.

Pueden ser creadas sesiones múltiples y pueden existir para un solo usuario de oracle; por ejemplo, un usuario con su nombre de usuario/contraseña (SYSTEM/MANAGER) puede conectarse en diferentes tiempos a la misma instancia oracle.

Cuando el servidor multitarea no está en uso, un server process es creado sobre el nombre de cada sesión de usuario; caso contrario, un solo server process puede ser compartido por muchas sesiones de usuario.

Oracle usando combinaciones de procesos user/server

Esta configuración de oracle esta permitida solamente en sistemas operativos que puedan mantener una separación entre la aplicación de la base de datos y el código de oracle en procesos sencillos. Esta separación es requerida para la integridad de los datos. El program interface es el responsable de la separación y protección del código de oracle y es responsable de pasar información entre la aplicación de base de datos y el programa de usuario oracle. Solamente una conexión esta permitida en algún momento. (Ver **figura 4**)

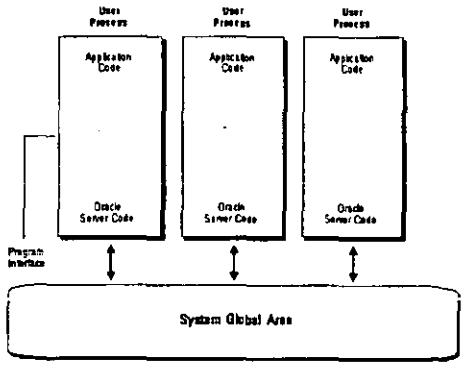


Fig. 4 En esta configuración la aplicación y el oracle server code corren todos en el mismo proceso.

Oracle usando procesos de servidor dedicado

En esta configuración cada proceso de usuario conectado a oracle tiene su correspondiente proceso de servidor dedicado. Además esta es una proporción uno a uno entre el numero de proceso de usuario y proceso de servidor. Una arquitectura de servidor dedicado permite a la aplicación cliente comenzar a ejecutar sobre un cliente comunicarse con otras computadoras que estén corriendo oracle a través de la red. (Ver **figura 5**).

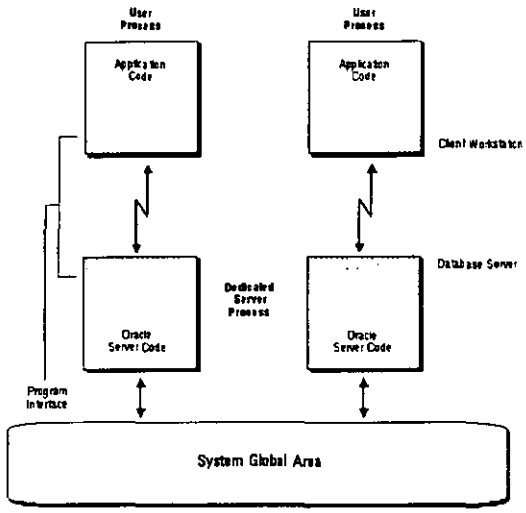


Fig. 5.En esta configuración un proceso de usuario ejecuta la aplicación de base de datos en una máquina y el server process en otra máquina, los dos procesos están separados.

El servidor multitarea

La configuración de servidor multitarea permite que muchos procesos de usuario puedan ser compartidos a través de un proceso dispatcher. El dispatcher rutea los requerimientos del cliente para el siguiente servidor de procesos compartido. La ventaja de una configuración de servidor multitarea es que el sistema es reducido de tal suerte que puede soportar un mayor número de usuarios. (Ver **figura 6**)

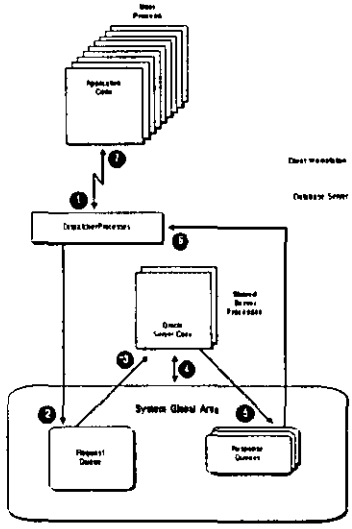


Fig. 6.

Ejemplo de configuración de oracle usando procesos de servidor dedicados

1. Una computadora que está corriendo actualmente un servidor de base de datos oracle usando diferentes procesos background
2. Una estación de trabajo (cliente) corre una aplicación de bases de datos (en un proceso de usuario) tal como SQL *Plus. La aplicación cliente intenta establecer una conexión con el servidor a través de un controlador de SQL Net .
3. El database server está corriendo actualmente su propio controlador SQL Net. El proceso Listener en la base de datos detecta la conexión requerida para la aplicación cliente y crea un proceso de servidor dedicado en el servidor de base de datos con el nombre de un proceso de usuario.
4. El usuario ejecuta un comando SQL simple. Por ejemplo el usuario inserta un renglón en una tabla.
5. El proceso de servidor dedicado recibe el comando. En este punto, dos caminos pueden seguirse para continuar el parseo del comando SQL.

    - Si el shared pool contiene una shared SQL area para comandos idénticos, el server process puede usar el area existente para ejecutar el comando del cliente.
    - Si el shared pool no contiene el shared SQL area para comandos idénticos, una nueva shared SQL area se asigna para el comando en el shared pool.

6. El server process recupera bloques de datos desde el actual datafile si es necesario, o bien, usa bloques de datos que ya estén almacenados en el buffer caché dentro del SGA de la instancia.
7. El server process ejecuta el comando SQL almacenado en el shared SQL area. Los datos son primero cambiados en el SGA . Después son escritos permanentemente en el disco a través del proceso DBWR una vez que se ha determinado la forma más eficiente para hacerlo. El proceso LGWR registra la transacción en un redo log file con estatus online solamente si el usuario hace una requisición de commit.
8. Si el requerimiento tiene éxito, el servidor envía un mensaje a través de la red al usuario; caso contrario un mensaje de error apropiado será transmitido a este.
9. En todo el procedimiento, los otros procesos background están corriendo y esperando alguna condición que requiera su intervención. Además, oracle maneja otras transacciones y previene contención entre las diferentes transacciones que está ejecutando.

Ejemplo de oracle usando el servidor multitarea

1. Un servidor de base de datos está corriendo oracle usando le configuración de servidor multitarea.
2. Una estación de trabajo (cliente) corre una aplicación de bases de datos (en un proceso de usuario) tal como SQL *Plus. La aplicación cliente intenta establecer una conexión con el servidor un controlador de SQL Net .
3. El servidor de base de datos está corriendo un controlador SQL Net. El proceso listener en el servidor de bases de datos detecta la conexión requerida por un proceso de usuario y determina cómo este proceso se reconecta usando la dirección de un dispatcher disponible.
4. El usuario usa un comando SQL simple. Por ejemplo, la actualización de un registro en una tabla.
5. El proceso dispatcher pone el proceso requerido en la cola la cual está en el SGA y compartida por el proceso dipatcher.
6. El proceso shared server que esté disponible checa los requerimientos comunes del dispatcher en la cola y recoge el siguiente comando SQL en la cola.
7. Solo un proceso shared server termina el procesamiento del comando SQL, el proceso muestra el resultado a la cola responsable de el dipatcher que envió el requerimiento.
8. El proceso dispatcher checa la cola responsable y envía el requerimiento completo al proceso de usuario que hizo el requerimiento.

OPERACION BASICA DE LA BASE DE DATOS

Levantar y dar de baja una instancia

Una base de datos Oracle puede no estar disponible para todos los usuarios. El administrador de la base de datos puede "levantar" la base de datos solamente si esta está abierta. Cuando una base de datos está abierta, los usuarios pueden accesar a la información que esta contiene. También bajo la misma circunstancia, el administrador de la base de datos (DBA) puede dar de baja la base de datos(DB), entonces esta quedará cerrada y los usuarios no podrán accesar más a la información que esta contiene.

Solamente un DBA puede abrir o cerrar la DB , un usuario común no tiene control sobre el estatus actual de una DB oracle. Por seguridad, la DB solo puede ser iniciada y dada de baja via conexión a Oracle con privilegios de administración.

Dependiendo del sistema operativo, uno de los siguientes prerequisitos es requerido para conectarse a oracle con privilegio de administrador:

- El usuario del sistema operativo debe contar con privilegios de sistema que le permitan conectarse usando privilegios de administrador.
- El usuario debe tener privilegios de SYSDBA o SYSOPER y la base de datos usa una contraseña para autentificar al DBA.
- El usuario debe conocer la contraseña para poder conectarse.

Además, los usuarios que pueden conectarse con privilegios de administrador solo pueden hacerlo en servidores dedicados y no en servidores compartidos.

Iniciar una instancia de la base de datos

Estos son los tres pasos para iniciar y hacer disponible una base de datos:

1. Iniciar una instancia.
2. Montar la base de datos
3. Abrir la base de datos.

Iniciar una instancia requiere generar un espacio para una SGA –una área de memoria compartida usada por la información de la DB y la creación de **Procesos Background** . Una instancia se debe iniciar antes de dar de alta la base de datos, ya que esta última se montará sobre la instancia. Si una instancia ha sido iniciada pero aun no montada, la base de datos aun no está asociada con esas estructuras de memoria y procesos.

Antes de que una instancia sea creada, oracle lee el parameter file, el cual determina la inicialización de la instancia. Este archivo incluye parámetros que controlan el tamaño del SGA, el nombre de la base de datos, a cual instancia se va a conectar, etc.

Se puede iniciar una instancia para alterarla en modo restringido, esto genera un límite de conexiones y solamente aquellos usuarios que tengan permiso de RESTRICTED SESSION, podrán hacer acceso a esta.

En circunstancias inusuales, una instancia no puede ser dada de baja "limpiamente", por ejemplo, uno de los procesos de la instancia no puede ser "matado". En tal situación, la base de datos podría generar un error durante un inicio normal de la instancia,. Para resolver este problema, el administrador de la base de datos debe matar todos los procesos de la instancia y levantarla nuevamente

Montando una base de datos

Después de iniciar la instancia, la base de datos es montada; esta permanece cerrada y es accesible solo para el administrador de la base de datos. El DBA puede necesitar iniciar una instancia y solamente montar la base de datos para completar tareas especificas de mantenimiento.

Cuando una instancia monta una base de datos, la instancia encuentra los control files y abre estos. Los control files son especificados en el parámetro CONTROL_FILES y este parámetro es usado para iniciar la instancia. Solamente es usado un solo control file para una base de datos, Oracle lee de allí el nombre de la DB , los datafiles y redo log files.

Modos para montar una base de datos

Si oracle permite que múltiples instancias sean montadas en la misma base de datos, consecuentemente, el DBA puede elegir entre correr la base de datos en modo exclusivo o en modo paralelo.

Si la primera instancia que se monta en la base de datos no está en modo exclusivo, solamente esa instancia puede montar la base de datos. En algunas versiones de Oracle, no se soporta el modo paralelo y solamente una instancia es montada en la base de datos en modo exclusivo.

Si al primer instancia que se monta en la base de datos es iniciada en modo paralelo (También llamado modo compartido), otras instancias que montan la base de datos son iniciadas en modo paralelo y pueden también montar la base de datos. El número de instancias que pueden montar una base de datos está sujeto a un número predeterminado máximo.

Abrir la base de datos

Abrir y montar la base de datos hace a esta disponible para una operación normal. Algunos usuarios válidos se pueden conectar a la base de datos y acccesar esta información solamente si esta abierta. Generalmente el DBA abre la base de datos y hace que esta esté disponible para su uso en general.

Cuando se abre una base de datos, Oracle abre los datafiles y los redo log files. Si el tablespace está en offline cuando la base de datos previamente se cerró, el tablespace y sus correspondientes datafiles serán puestos en offline cuando se vuelva a abrir la DB.

Si alguno de los datafiles o redo log files no están presentes cuando se inicie la base de datos, Oracle retornará un error. Se debe entonces ejecutar una recuperación de un respaldo antes de abrir de nueva cuenta la base de datos

Recuperación de la Instancia

Si la DB ha sido dada de baja ya sea por que el DBA ha abortado la instancia o por una falla ocurrida cuando la base de datos estaba corriendo, oracle automáticamente ejecuta una recuperación de la instancia en donde la base de datos es vuelta a levantar.

Baja de la instancia y la base de datos

Existen tres pasos para dar de baja una instancia y una base de datos a la cual se está conectado.

1. Cerrar la base de datos
2. Desmontar la base de datos
3. Dar de baja la instancia.

Oracle automáticamente ejecuta todos los pasos cuando una instancia es dada de baja.

Cerrar una base de datos

El primer paso para dar de baja la base de datos es cerrar la DB. Cuando se cierra la base de datos, oracle guarda todos los datos en la base de datos y hace un recovery del SGA en los datafiles y redologfiles que se encuentren en línea (online). Algunos datafiles y algunos tablespaces que se encontraban fuera de línea(offline) han sido cerrados ya. Cuando se vuelve a abrir la base de datos el tablespace que está offline y sus datafiles permanecen offline y cerrados respectivamente. El control files permanece abierto y después que una base de datos es cerrado aun permanece montado.

Cerrar la base de datos y abortar la instancia

En alguna situación rara de emergencia, se puede abortar la instancia de una base de datos para cerrarla y completar la baja de la base instantáneamente. Este proceso es rápido por que la operación de escritura de todos los datos en los buffers del SGA para los datafiles y redo log files es saltada. La siguiente vez que se monte la base de datos requerirá una recuperación de la instancia la cual oracle ejecuta automáticamente.

Desmontar la base de datos

El segundo paso para completar la baja de la base de datos es desmontar o desasociar la base de datos de la instancia. Después de desmontar la base de datos, solamente una instancia permanece en la memoria de la computadora. Después que la base de datos es desmontada oracle cierra los control files de la base de datos.

Dar de baja la instancia

El paso final para dar de baja la base de datos es dar de baja la instancia. Cuando se da de baja la instancia, el SGA es removido de la memoria y los proceso background son concluidos.

En circunstancias inusuales, la instancia se puede dar de baja de una forma no muy limpia; todas las estructuras pueden ser no removidas de la memoria o alguno de los procesos background puede ser que no se dé de baja. Cuando quedan algunos objetos de otras instancias, la nueva instancia puede levantarse con errores. Para manejar este problema, el administrador de la base de datos deberá forzar la nueva instancia para que se inicie correctamente o se puede usar el comando SHUTDOWN ABORT.

El parameter file

Para iniciar una instancia, oracle debe leer un archivo de parámetros(parameter file). Un Parameter file es un archivo de texto que contiene una lista de la configuración de los parámetros. Se les debe asignar a estos parámetros un valor particular e iniciar muchos de estos en memoria.

Entre otras cosas el parameter file le dice a oracle lo siguiente:

- El nombre de la base de datos para la cual se levantará la instancia.
- Qué tanta memoria se usará para las estructuras del SGA
- Qué archivos de redo log files se usarán
- Cuál es el nombre y la localización de los control files de la base de datos.
- Los nombres de los segmentos de rollback en la DB.

El administrador de la base de datos puede ajustar las variables para mejorar el rendimiento de la base de datos.

Ejemplo de Parameter File

```
db_block_buffers = 550
db_name = ORA7PROD
db_domain = US.ACME.COM
#
```

```
license_max_users = 64
#
control_files = filename1, filename2
#
log_archive_dest = c:\logarch
log_archive_format = arch%S.ora
log_archive_start = TRUE
log_buffer = 64512
log_checkpoint_interval = 256000
# rollback_segments = rs_one, rs_two
```

ESTRUCTURAS DE LA BASE DE DATOS

Relación entre data blocks, extents y segments

Oracle asigna el espacio de base de datos para todos los datos. Las unidades lógicas de asignación son los data blocks, extents y segments en la **figura 7** podemos observar qué relación existe entre estas unidades lógicas, esto para su próximo entendimiento.

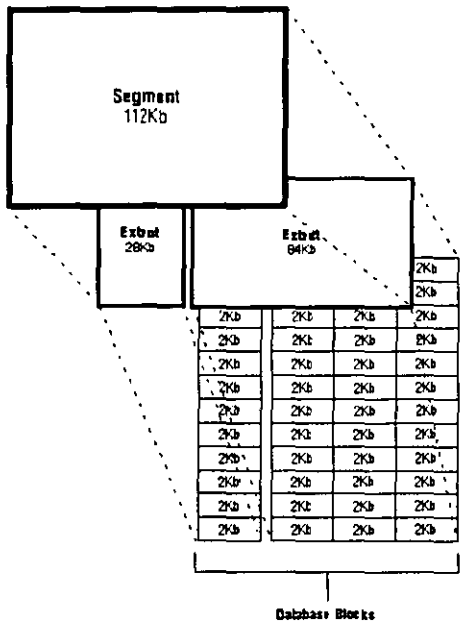


Fig. 7.

Un *data block* es el nivel más fino de oracle, los datos se almacenan en data blocks (también llamados bloques lógicos, bloques de oracle o páginas). Un data block corresponde a un número de bytes de espacio fisico en el disco. Cuando se crea una base de datos, se debe poner el tamaño de todos los bloques. Los data blocks son pequeñas unidades de almacenamiento que oracle usa para asignar. El tamaño de los data blocks deberia ser el de un múltiplo dependiendo del tamaño del sistema operativo dentro del máximo limite.

Oracle hace requerimientos de datos en múltiples datablocks no así en bloques del sistema operativo, sin embargo se podría poner el tamaño del bloque como un múltiplo del tamaño del sistema operativo para evitar accesos innecesarios (I/O)

El siguiente nivel lógico de espacio en la base de datos es el **extent**. Un extent es un número específico de data blocks contínuos y están asignados para almacenar un tipo específico de información.

El siguiente nivel de almacenamiento es el segment. Un segment es un grupo de extents que han sido asignados para un tipo especifico de estructura de datos y además todos son almacenados en el mismo tablespace. Por ejemplo, los datos de una tabla están almacenados en su propio segmento de datos, mientras que cada segmento de indices está almacenado en su propio segmento de índices.

Oracle asigna espacio para segments en extents, por consiguiente cuando los extents existentes de un segmentos están llenos, oracle asigna otro extent para el mismo segmento. Debido a que los extents son asignados conforme se requieren los extents de un segmento pueden o no pueden estar en forma continua en el disco.

En la **figura 8** se muestra en forma aun más clara esta relación de pertenencia entre los datablocks, extents y segments

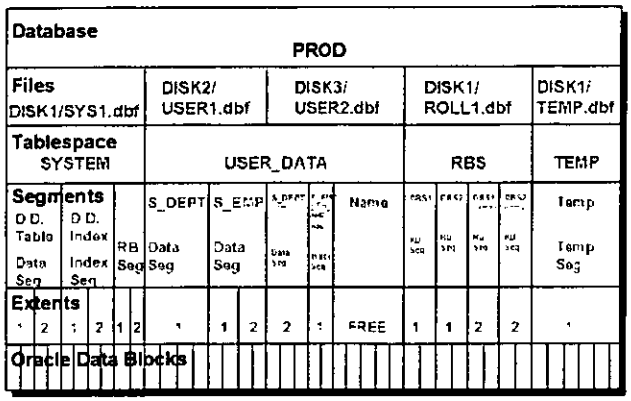


Fig. 8. Podemos observar la relación entre las estructuras de la base de datos, unas contenidas en otras y como último nivel los oracle data blocks.

Data Blocks

Oracle maneja el espacio de almacenamiento en los datafiles en unidades llamadas data blocks. Un data block es una pequeña unida de entrada/salida (I/O) usada por la base de datos.

El formato del oracle block es similar si el data block contiene tablas, índices o clusters. Para ver gráficamente la estructura de un data block podemos observar la **figura 9** en donde se muestran sus componentes.

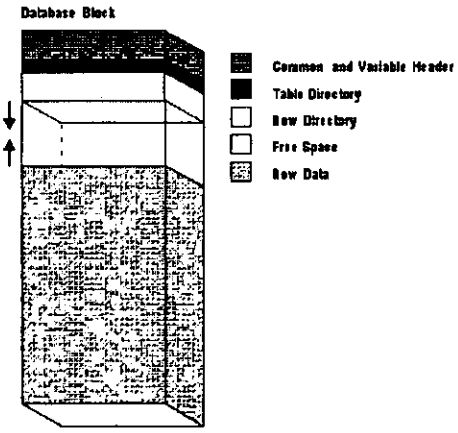


Fig. 9

La *cabezera o Hedaer* que es su nombre en inglés, contiene información general del bloque tal como la dirección y el tipo de segmento, por ejemplo, de datos, de índices o de rollback. Algunos bloques son de tamaño fijo por el tamaño del total de los bloques es variable en promedio entre 84 y 107 bytes.

El *table directory* es la porción del bloque que contiene información acerca del los renglones que contienen las tablas en ese bloque.

El *row directroy* es la porción del bloque que contiene información de los registros actuales en el bloque, incluyendo direcciones para cada registro y para cada parte del registro en el row data area.

El row data es la porción del bloque que contiene datos de tablas o de índices.

El espacio libre (*free space*) es usado para insertar nuevos registros y para hacer actualizaciones a los registros que requieren espacio adicional.

Un data block asignado por un data segment de un tabla, cluster o un index pueden también usar espacio para transacciones. Una transacción es requerida en un bloque para cada comando INSERT, UPDATE, DELETE y SELECT...FOR UPDATE accesando uno o más renglones en el block. El espacio requerido para las transacciones depende del sistema operativo, sin embargo una transacción en muchos sistemas operativos requiere aproximadamente de 23 bytes.

Existen dos parámetros para manejo de espacio, el PCTFREE y PCTUSED, estos permiten desarrollar el control de uso del espacio libre para insertar o para actualizar los registros en los data blocks, Se especifican esos parámetros solamente cuando se crean o alteran tablas o clusters (data segments). También se puede especificar el parámetro de almacenamiento PCTFREE cuando se crean o alteran índices (index segment).

Parámetro PCTFREE

El parámetro PCTFREE es usado para poner el porcentaje de bloque reservado(kept free) para hacer posibles actualizaciones en los registros que ya están contenidos en el bloque, pro ejemplo si asignamos 20 a este parámetro al crear una tabla significa que el 20% de cada data block usado para este data segment de la tabla deberá estar libre y disponible par posibles actualizaciones de los registros que existan ya dentro de cada bloque

Parámetro PCTUSED

Después de que un bloque se llena, como esta determinado por el PCTFREE, oracle no considera este bloque para la inserción de nuevos registros mientras el porcentaje del bloque no caiga abajo del PCTUSED. Antes, el valor es archivado y oracle usa el espacio libre del data block solamente para actualizar los registro que ya están contenidos allí. Por ejemplo, si especificamos PCTUSED 40 al crear una tabla, el data block usado por este segmento de datos de tabla no es considerado para la inserción de algún nuevo registro mientras la cantidad de espacio usado no caiga abajo del 39% o menos.

PCTFREE y PCTUSED trabajan juntos para optimizar la utilización del espacio en los data blocks de los extents dentro del data segment. En una nueva asignación de un data block, el espacio disponible para inserciones es el tamaño del bloque menos la suma del bloque y PCTFREE. Las actualizaciones para datos existentes pueden usar cualquier espacio disponible en el bloque, así mismo, las actualizaciones pueden reducir el espacio disponible en el bloque para que sean menores que PCTFREE espacio reservado para actualizaciones pero no accesibles para inserts.

Para cada segmento de datos o índices, oracle tiene una o más free list; una free list es una lista de data blocks que han sido asignadas para que los extents de los segmentos tengan un mayor espacio que PCTFREE, estos bloques están disponibles para inserts. Cuando se usa un comando INSERT, oracle checa una free list de la tabla para el primer bloque disponible y darle uso si es posible; si el espacio libre en el bloque no es lo suficientemente grande para acomodar el comando INSERT, y este es menor que PCTUSED, oracle toma otro bloque de la free list. El uso de múltiples free list por segmento puede reducir la contención para free list cuando se hacen inserts concurrentemente.

Extents

Un extent es una unidad lógica de almacenamiento en la base de datos, un espacio de asignación hecho de un número contiguo de data blocks. Cada segmento está compuesto de uno o más extents. Cuando el espacio existente en un segment está usado por completo, oracle asigna un nuevo extent para el segment.

No importa de que tipo, cada segmento en una base de datos es creado con al menos un extent para mantener los datos. Este extent es llamado el extent inicial del segment.

Por ejemplo, cuando se crea una tabla el data segment contiene un extent inicial de un número específico de data blocks. Asimismo no pueden insertarse nuevos registros aun, los data blocks de oracle que correspondan al extent inicial son reservados para los registros de esa tabla.

Si el datablock de un extent inicial de un segmento llega a estar lleno y requiere de más espacio para almacenar datos nuevos, oracle automáticamente asigna un extent para ese segmento. Para mantener en orden los propósitos del extent, cada segment en la base de datos contiene un bloque de cabezera que describe las caracteristicas del mismo.

Oracle controla la asignación de los extents para un segment dado. El procedimiento para asignar un nuevo extent para un segment es el siguiente:

1. Oracle busca espacio libre (dentro del tablespace que contiene al segment) y encuentra el espacio libre para el nuevo extent y usa el siguiente algoritmo.

   1.1 Oracle busca un conjunto de data blocks contiguos que generen un nuevo extent y agrega un bloque para reducir la fragmentación interna. Por ejemplo, si un nuevo extent requiere 19 data blocks, oracle busca exactamente 20 data blocks contiguos, sin embargo, si el nuevo extent tiene menos de 5 bloques, oracle no puede agregar un bloque extra para este requerimiento.

   1.2 Si el número exacto requerido no es encontrado, oracle entonces busca un grupo contiguo de data blocks igual o más grande que la cantidad necesitada. Si oracle encuentra un grupo de bloques contiguos que sea menor a cinco bloques se despliega un grupo de bloques en un extent separado el cual es del tamaño necesario; Si oracle encuentra un grupo de bloques que es más grande al tamaño que se necesita, pero menor de cinco bloques, este se asigna todo al de bloques contiguos.

   Continuando con el ejemplo si oracle no encuentra un grupo de exactamente 20 data blocks contiguos, entonces busca un grupo contiguo de data blocks mayor que 20, si en la primer búsqueda oracle encuentra 25 o más bloques, este rompe el bloque y asigna 20 al nuevo extent, de otra forma, asigna todos los bloques (en un rango de 21 a 24) al nuevo extent.

   1.3 Si oracle no encuentra un conjunto de data blocks contiguos, entonces une data blocks adyacentes en su correspondiente tablespace tal que el tamaño de los data blocks contiguos lo formen. (El proceso SMON periódicamente une espacio libre adyacente). Después de unir los data blocks de un tablespace, oracle ejecuta nuevamente su búsqueda. Si un extent no puede ser asignado después de una segunda búsqueda, se retorna un error desde oracle.

2. Solamente si oracle encuentra el espacio necesario libre en el tablespace, asigna una porción de espacio libre que corresponda al tamaño del extent.
3. Oracle actualiza la cabecera del segmento y el diccionario de datos para mostrar que un nuevo extent ha sido asignado y que la asignación del espacio no es más grande que el espacio libre.

Generalmente oracle limpia los bloques para una nueva asignación de extents cuando un extent es usado por primera vez. En algunos casos sin embargo cuando el DBA usa ALTER TABLE o ALTER CLUSTER con la opción ALLOCATE EXTENT mientras se usa la free list, oracle limpia los bloques del extent cuando este asigna el extent.

Todos los extents asignados a un index segment están asignados tan grandes como indices existan. Cuando se borra un índice o cluster, oracle hace un reclamo del extent para otros usos dentro del tablespace.

Oracle checa periódicamente para ver si existen segmentos de rollback de la base de datos que hubieran crecido más de su tamaño óptimo. Si un segmento de rollback es más grande que el óptimo, automáticamente se desasigna uno o más extents del segmento de rollback.

Cuando oracle completa la ejecución de un comando requerido por un segmento temporal, automáticamente se elimina el segmento y retorna los extents asignados al segment de el tablespace asociado.

Segments

Un segment es un grupo de extents que contienen todos los datos de una específica estructura de almacenamiento dentro del tablespace. Por ejemplo, para cada tabla, oracle asigna uno o más extents para el segmento de datos de esa tabla y para cada índice se asigna uno o más extents para ese segmento de índices.

Existen cuatro tipos de segmentos usados en las bases de datos de oracle:

- Data segments
- Index segments
- Rollback segments
- Temporary segments

Data segments

Todas las tablas (incluyendo snapshots y snapshot logs) en una base de datos oracle tienen un data segment para mantener todo lo relacionado a esa tabla. Oracle crea estos data segment cuando se crea un objeto con los comandos CREATE TABLE/SNAPSHOT/SNAPSHOT LOG. Cualquier cluster en una base de datos oracle usa un data segment para mantener los datos de todas sus tablas. Se crea un data segment para el cluster cuando se usa el comando CREATE CLUSTER. El parámetro de almacenamiento para una tabla, snapshot, snapshot log o cluster controlan el camino en que estos data segment serán asignados.

Index segment

Todos los índices en una base de datos oracle tienen un index segment para mantener todos sus datos. Se crea un index segment para el índice cuando se usa el comando CREATE INDEX. Este comando permite especificar el parámetro de almacenamiento para los extents de el index segment y el tablespace en el cual se creara el index segment (Los segmentos de una tabla y un índice no tienen que ocupar el mismo tablespace) .

Rollback segments

Cada base de datos contiene uno o más rollback segments. Un rollback segment es una porción de la base de datos que registra las acciones de las transacciones si esa transacción puede ser invertida o deshecha. Los rollback segment son usados para proveer consistencia en la lectura, para deshacer transacciones y para recuperación de la base de datos.

La información en un rollback segment consiste en diferentes *entradas rollback*. Una entrada de rollback incluye bloques de información (el nombre y ID correspondiente al dato que fue cambiado) y los datos como estos eran antes de la operación de la transacción. Oracle liga las entradas de rollback para la misma transacción tal que las entradas pueda ser fácilmente encontradas si fuese necesario para una transacción rollback. Los usuarios de la base de datos o los DBA no pueden accesar o leer los rollback segments; solamente oracle puede escribir o leer de ellos.

Debido a que las entradas rollback cambian los data blocks, oracle también registra los cambios de estos en los redo log files. Este segundo registro de información de rollback es muy importante para transacciones activas que aun no has sido guardadas en el

momento en el que el sistema se caiga. Si ocurre una caída del sistema, oracle automáticamente recupera la información de los segmentos de rollback, incluyendo las entradas rollback de las transacciones activas como parte de una instancia o media recovery[5]. Oracle ejecuta rollbacks de las transacciones que no fueron guardadas o deshechas en el momento de la falla después de que la recuperación fue completada.

Oracle mantiene una transacción para cada rollback segment contenido en la base de datos. Cada tabla es una lista de todas las transacciones que usa el rollback segment asociado y las entradas rollback para cada cambio ejecutado por esa transacción. Oracle usa las entradas rollback en un rollback segment para ejecutar una transacción rollback y para crear consistencia en la lectura de los datos recuperados por las consultas (querys).

Cada que comienza una transacción de usuario, oracle asigna la transacción a un Rollback segment:

Oracle puede asignar la transacción automáticamente al siguiente rollback segment disponible. La asignación de la transacción ocurre cuando se usa el primer comando DML o DDL[6] en la transacción. Oracle nunca asigna transacciones de solo lectura (recuperadas solamente por querys) a un rollback segment.

Una aplicación puede asignar una transacción a un rollback segment específico. Al inicio de la transacción un desarrollador o usuario puede especificar un especifico rollback segment que oracle deberá usar cuando se ejecute la transacción. Esto permite al desarrollador seleccionar un rollback segment grande o pequeño que sea el más apropiado para esa transacción. Mientras dure la transacción, el proceso de usuario asociado escribe información de rollback solamente en el rollback segment asignado.

Cuando se hace commit a una transacción , oracle actualiza la información pero esta no es destruida inmediatamente. Cada rollback segment puede manejar un número de transacciones para una instancia. A menos que se especifique una asignación de una transacción a un rollback segment en particular, oracle distribuye las transacciones activas entre los rollback segment disponibles tal como todos los rollback segments están asignados aproximadamente al mismo número de transacciones activas. La distribución no depende sobre el tamaño del rollback segment disponible, empero, en entornos donde las transacciones generan la misma cantidad de información rollback, todos los rollback segments pueden ser del mismo tamaño.

Cuando se crea un rollback segment, se puede especificar el parámetro de almacenamiento para la asignación de los extents para este segmento. Cada rollback segment debe tener al menos dos extents asignados. Una transacción escribe secuencialmente en un simple rollback segment; cada transacción escribe solamente en un extent de un rollback segment en un tiempo dado. De la misma forma muchas transacciones activas (transacciones en progreso a las que aun no se les hace commit o rollback) pueden escribir concurrentemente en un simple rollback segment, en el mismo extent de un rollback segment; sin embargo, cada bloque de un extent en rollback segment puede contener información solamente de una transacción simple .

Cuando una transacción corre fuera del espacio en el extent actual y necesita continuar escribiendo, oracle debe encontrar un extent disponible dentro del mismo rollback segment en el cual se está escribiendo. Para realizar esta tarea se tienen dos opciones:

- Se puede reusar un extent ya asignado para el rollback segment
- Se puede adquirir y asignar un nuevo extent para el rollback segment.
-

La primer transacción que necesita adquirir más espacio de rollback checa el siguiente extent del rollback segment. Si el siguiente extent del rollback segmento contiene información activa para deshacer, oracle hace de este el extent actual y todas las transacciones que necesitan más espacio pueden entonces escribir sobre este.

La **figura 10** muestra dos transacciones T1 y T2, las cuales continuan escribiendo desde el primer extent de un rollback segment.

[5] Un media recovery recupera los archivos de una base de datos donde la información corresponda al más reciente respaldo que se tenga antes que el disco haya fallado.

[6] Lenguaje de manipúlación de datos(DML), usa comandos tales como insert, update, delte, etc. También esta el Lenguaje de definición(DDL) de datos que cuenta con comandos como create y drop

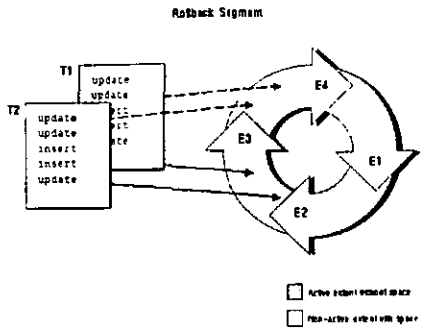


Fig. 10

Como las transacciones continúan escribiendo hasta llenar el extent actual, oracle checa el siguiente extent asignado por el rollback segment para determinar si este está disponible. Continuando con el ejemplo, en la **figura 11** cuando E4 está completamente lleno, T1 y T2 continúan escribiendo en el siguiente extent asignado al rollback segment que está disponible; E1 es esté extent. Aquí se muestra el ciclo natural de un extent usado en un rollback segment.

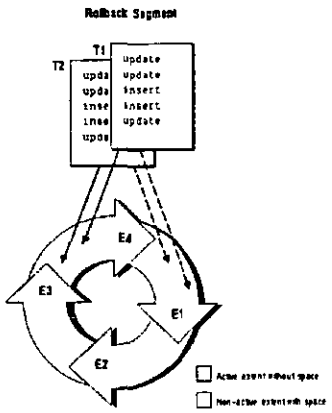


Fig. 11

Para continuar escribiendo información Rollback para una transacción, oracle siempre intenta reusar el siguiente extent del primer anillo. Sin embargo, si el siguiente extent contiene datos activos, oracle debe asignar un nuevo extent. Oracle puede asignar un nuevo extent para un rollback segment mientras el número de extents no alcance el valor del parámetro de almacenamiento MAXEXTENTS.

En la misma **figura 11**, cuando E4 está completamente lleno, las transacciones no pueden continuar escribiendo en el siguiente extent de la secuencia (E1) por que este contiene entradas rollback activas, debido a esto, oracle asigna un nuevo extent E5 a este rollback segment y las transacciones se continúan escribiendo en este nuevo extent.

Cuando se crea o se altera un Rollback segment, se puede usar el parámetro OPTIMAL, el cual aplica solamente para rollback segments y se especifica en bytes. Si una transacción necesita continuar escribiendo información rollback de un extent para otro extent en el rollback segment, oracle compara el tamaño actual de el rollback segment; si el rollback segment es más grande que su tamaño óptimo, oracle lo desasigna mientras el tamaño total del rollback segment es igual al del óptimo.

Si se le hace un drop a un rollback segment (borrar un rollback segment), oracle retorna todos los extents del rollback segment para su tablespace, los extents retornados estarán entonces disponibles para otros segmentos en el tablespace.

Tipos de segmentos de rollback

Cuando una instancia abre una base de datos, esta debe adquirir uno o más rollback segments de tal forma que la instancia pueda manejar información rollback producida por las subsecuentes transacciones. Una instancia puede tener rollback segments privado o públicos. Un rollback segment privado es adquirido explícitamente por una instancia cuando esta abre la base de datos. Un rollback segment público de un conjunto de rollback segments que alguna instancia requiere y que puede usarse. Cualquier número de rollback segment privados y públicos puede existir en la base de datos. Como una instancia abre una base de datos, la instancia intenta adquirir uno o más Rollback segments de acuerdo a las siguientes reglas:

La instancia debe adquirir al menos un rollback segment. Si la instancia es la única en accesar ala base de datos, esta debe adquirir el SYSTEM segment; si la instancia es una de tantas instancias accesando a la base de datos, esta debe adquirir el SYSTEM rollback segment y al menos otro rollback segment, de otra manera, oracle retorna un error y la instancia no puede abrir la base de datos.

La instancia primero intenta adquirir todos los rollback segment privados especificados por el parámetro ROLLBACK_SEGMENTS. Si una instancia abre la base de datos e intenta adquirir un rollback segment privado ya reclamado por otra instancia, la segunda instancia que intentará adquirir el rollback segment recibirá un mensaje de error durante el inicio. También se muestra un error si una instancia intenta adquirir un rollback segment privado que no existe.

Si la instancia ya ha adquirido un rollback segment privado, no son requeridas más acciones, sin embargo, si la instancia requiere más segmentos de rollback entonces intentará adquirir rollback segments públicos

Rollback Segment SYSTEM

Oracle crea un rollback segment inicial llamado SYSTEM siempre que una base de datos es creada. Este segmento está en el tablespace de SYSTEM y usa los parámetro de almacenamiento de este mismo. No se puede eliminar el rollback segment de SYSTEM. Una instancia siempre adquiere el rollback segment SYSTEM adicionalmente a algún otro segmento necesario.

Si existen múltiples rollback segments, oracle intenta usar el rollback segment de SYSTEM solamente para transacciones especiales de system. En general, después de la creación de la base de datos se debería crear también al menos un rollback segment adicional en el tablespace de system

Estatus de los segmentos de rollback

Un rollback segment está siempre en uno de distintos estatus, dependiendo si este está offline, adquirido por una instancia, inmerso en una transacción sin resolver, en una recuperación necesaria o borrado. El estatus de los rollback segments determina cuando este puede ser usado en transacciones. Los estatus de los rollback segments son los siguientes

OFFLINE. No ha sido adquirido por una instancia.
ONLINE. Ha sido adquirido por una instancia y puede contener datos de transacciones activas.
NEEDS RECOVERY. Contiene datos de transacciones sin commit que no pueden ser deshechos (por que los datafiles fallaron).
PARTLY AVAILABLE. Contienen datos de transacciones distribuidas sin solución.
INVALID. Ha sido eliminado.

En la **figura 12** se muestra como los rollback segments cambian sus estatus.

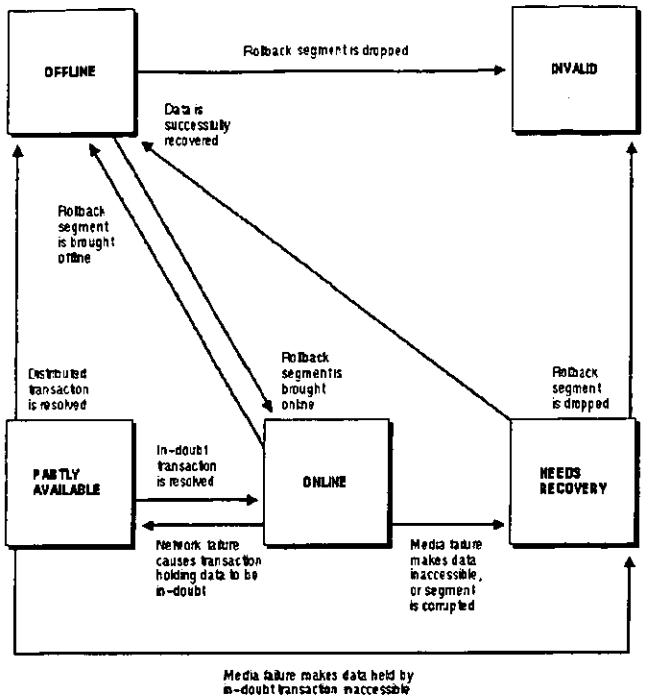


Fig. 12

La tabla del diccionario de datos DBA_ROLLBACK_SEGS contiene el estatus de cada rollback segment . Cuando un rollback segment se pone en offline de tal forma que las transacciones no puedan ser deshechas de manera inmediata, oracle escribe un rollback segment diferido que contendrá las entradas rollback que no podrán ser aplicadas en el tablespace, de esta forma, podrán ser aplicadas cuando lleguen a estar en online. Estos segmentos desaparecen tan pronto como el tablespace se recupera. Oracle crea automáticamente rollback segments diferidos en el tablespace de SYSTEM.

Temporary Segments

Cuando se procesan querys, oracle comúnmente requiere áreas de trabajo temporales para procesar comandos SQL, automáticamente se asignan estos espacios del disco llamados temporary segments, generalmente oracle requiere de un temporary segment como un área de trabajo para ordenamiento. Oracle no crea un segmento si la operación de ordenamiento puede ser hecha en memoria o si se encuentra algún otro camino para ejecutar la operación usando índices.

Los siguientes comandos pueden requerir el uso de temporary segments:

- CREATE INDEX
- SELECT ... ORDER BY
- SELECT DISTINCT ...
- SELECT ... GROUP BY
- SELECT ... UNION
- SELECT ... INTERSECT
- SELECT ... MINUS
- unindexed joins
- certain correlated subqueries

Por ejemplo, si un query contiene una cláusula DISTINCT, un GROUP BY o un ORDER BY, oracle puede requerir más de un temporary segment. Si las aplicaciones usan comúnmente comandos de la lista anterior, el DBA debe mejorar el rendimiento ajustando el parámetro de inicialización SORT_AREA_SIZE.

Oracle asigna temporary segment como vaya siendo necesario durante una sesión de usuario, por ejemplo, un usuario puede emitir un query y requerir tres temporary segments. Oracle elimina estos temporary segments cuando el comando ha completado su operación, se crean estos en el temporary tablespace en el que el usuario usa el comando. Se especifica el temporary tablespace cuando se crea al usuario con el comando CREATE USER o ALTER USER usando la opción TEMPORARY TABLESPACE, de otra manera, el tablespace de default es el de SYSTEM.

Tablespaces y Datafiles

Oracle almacena datos en forma lógica en los tablespace y en forma física en los datafiles asociados a su correspondiente tablespace, en la **figura 13** se muestra la relación entre los tablespaces y los datafiles.

Aunque la base de datos, los tablespace, datafiles y segments están estrechamente relacionados, estos tienen diferencias importantes :

Databases y Tablespaces
Una base de datos oracle está compuesta por uno o más unidades de estructuras lógicas llamadas *tablespaces*. Los datos de la base de datos (*database*) están almacenados colectivamente en los *tablespaces* de la base de datos.

Tablespaces y Datafiles
Cada *tablespace* en una base de datos oracle esta compuesto de uno o mas archivos de sistema operativo llamados *datafiles*. Un *datafile* de un *tablespace* almacena los datos físicamente en el disco.

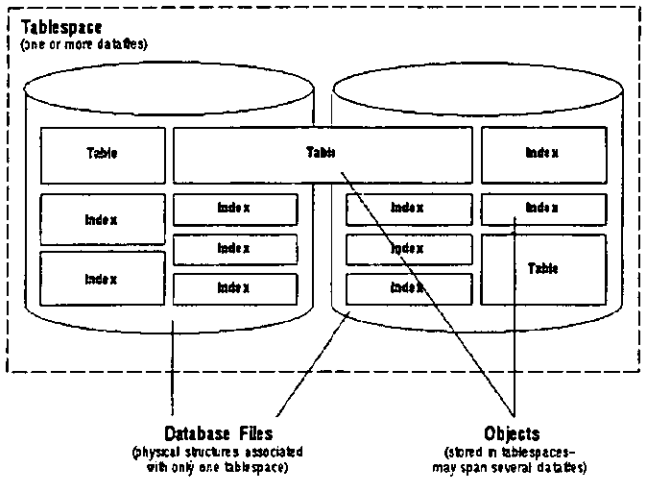


Fig. 13.

Databases y Datafiles
Los datos en una base de datos están almacenados en forma colectiva en los *datafiles* por los que está formado cada *tablespace* de la *database*. Por ejemplo, una base de datos simple de oracle podria contener un *tablespace* y un *datafile*. Una base de datos más complicada podría contener tres *tablespaces* cada uno compuesto por dos *datafiles*.

Objetos del schema, segments, y tablespaces.
Cuando un objeto del schema tal como una tabla o índice es creado, un segmento es creado dentro de un tablespace designado en la base de datos. Por ejemplo, si se crea una tabla en un tablespace especifico usando el comando CREATE TABLE con la opción TABLESPACE, oracle asigna el espacio para los data segment en uno o más datafiles que constituyen el tablespace especificado. Los objetos en el segmento asignan espacio solamente en un tablespace de la base de datos.

Arquitectura de almacenamiento Oracle

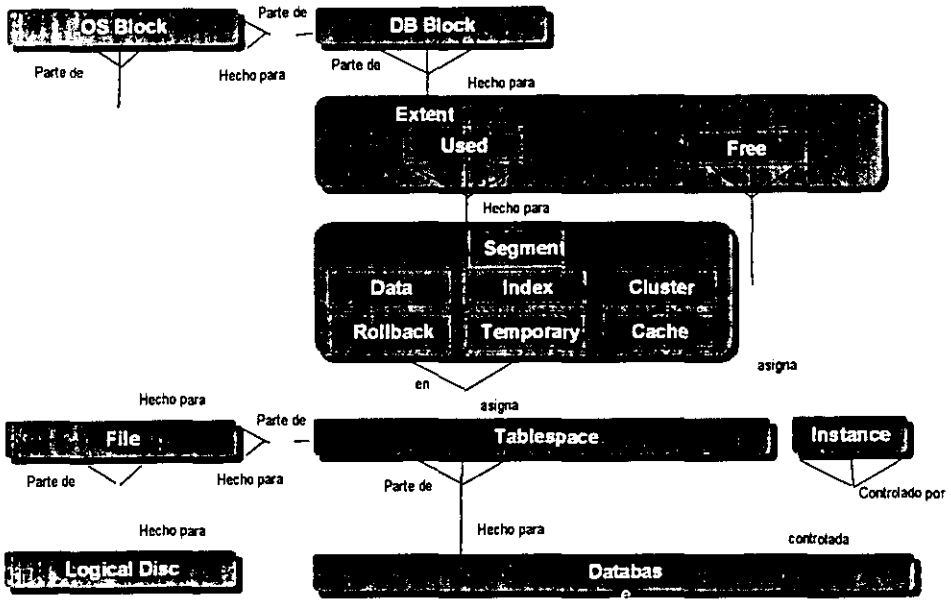


Fig. 14. En esta figura se representa la relación existente entre las diferentes estructuras que componen la base de datos oracle.

Tablespaces

Una base de datos está dividida en una o más unidades lógicas de almacenamiento llamadas tablespaces. Un DBA puede usar tablespaces para realizar las siguientes tareas:

- Controlar la asignación de espacio para los datos en la base de datos
- Asignar cuotas especificas de espacio para los usuarios de la base de datos
- Controlar la disponibilidad de los datos a través de generar tablespaces individuales con estatus online u offline
- Ejecutar respaldos parciales de la base de datos u operaciones de recuperación
- Asignar el almacenamiento de datos a través de el mejoramiento del performance.

Un DBA puede crear tablespaces nuevos, agregar y remover datafiles para esos tablespaces, fijar o alterar los segmentos, hacer un tablespace solo de lectura o lectura escritura, hacer un tablespace temporal o permanente y borrar tablespaces.

El SYSTEM Tablespace

Toda base de datos oracle contiene un tablespace llamado SYSTEM que se genera automáticamente cuando la base de datos es creada. El tablespace SYSTEM siempre contiene las tablas del diccionario de datos de toda la base de datos. Una base de datos pequeña podría necesitar solamente el tablespace SYSTEM; empero, se recomienda crear al menos un tablespace adicional para almacenar datos de usuario separados de la información del diccionario de datos, esto permitiría más flexibilidad en las operaciones de administración y reduciría en gran mediada la contención entre los objetos del diccionario y los objetos del schema para los mismos datafiles. El tablespace de SYSTEM siempre tiene el estatus de ONLINE. Todos los datos almacenados como program units de PL/SQL (procedures, functions, packages y triggers) residen en el tablespace de SYSTEM.

Asignación de espacio para la base de datos

Para ampliar una base de datos se tienen tres diferentes opciones. Se puede agregar otro datafile para un tablespace existente, por eso se incrementa la cantidad de espacio asignado en el disco para el correspondiente tablespace. La **figura 15** muestra el espacio incrementado de la base de datos.

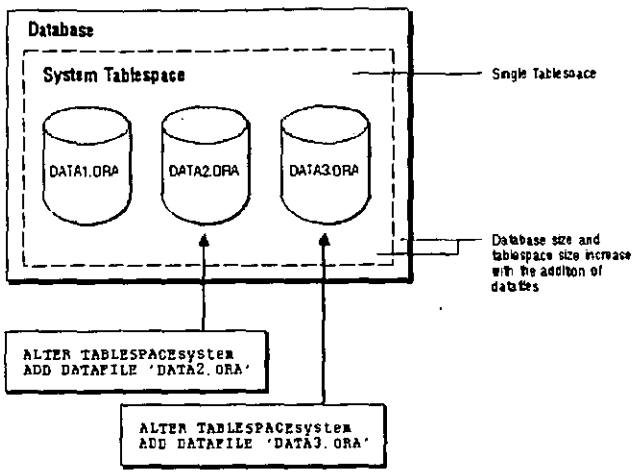


Fig. 15.

En forma alternativa, el DBA puede crear un tablespace nuevo (definido por un datafile adicional) para incrementar el tamaño de la base de datos. **(Fig. 16)**

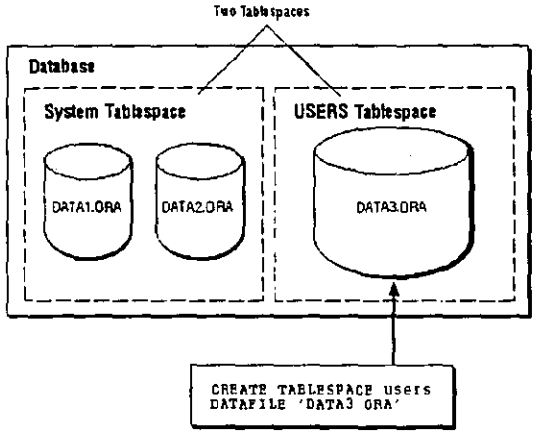


Fig. 16

El tamaño de un tablesapce es el tamaño de los datafiles que constituyen el tablespace y el tamaño de la base de datos es la suma de todos los tablespaces que conforman la base de datos.

La tercera opción es cambiar el tamaño de los datafiles o permitir que los tablespaces existentes crezcan dinámicamente conforme el espacio vaya siendo necesario. Para lograr esto, se pueden alterar los archivos existentes o bien adicionar archivos dinámicamente. **(figura 17)**

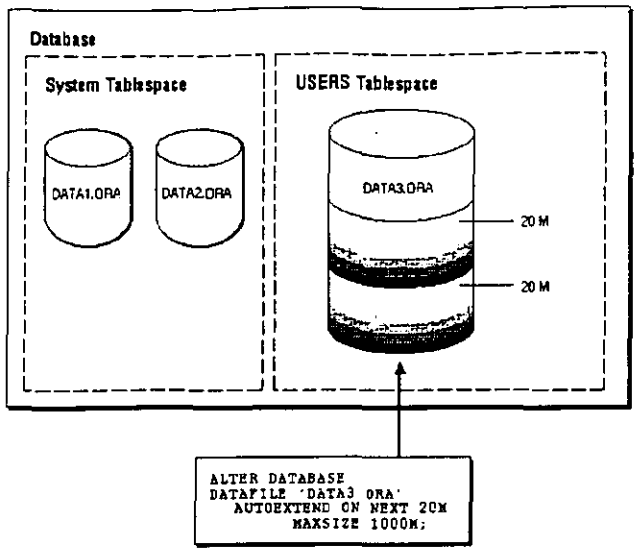


Fig. 17 Aumentando el tamaño de la base de datos a través de datafiles que crecen dinámicamente.

Tablespaces Online y Offline

Un DBA puede traer cualquier tablespace (excepto el tablespace SYSTEM) en una base de datos oracle accesible o no accesible siempre y cuando la base de datos esté abierta. El tablespace SYSTEM siempre debe estar online por que el diccionario de datos siempre debe estar disponible para oracle

Un tablespace está normalmente online de manera que los datos contenidos en este estén disponibles para los usuarios de la base de datos, sin embargo, el DBA debe tener un tablespace offline por alguna de las siguientes razones:

- Para hacer que una parte de la base de datos no esté disponible mientras se permite el acceso normal para el resto de la base de datos.
- Para ejecutar un respaldo de la base de datos
- Para hacer una aplicación en este grupo de tablas temporales no disponible mientras se actualiza o da mantenimiento a la aplicación.

Cuando un tablespace está en offline, oracle no permite que algún comando SQL haga referencia a objetos contenidos en el tablespace. Las transacciones activas que han completado algún comando que hace referencia a los datos dentro del tablespace que ha sido puesto en offline no deben afectar el nivel de transacción, oracle salva los datos rollback correspondiente a los comandos que afectan datos en el tablespace offline en un deferred rollback segment (en el tablespace SYSTEM). Cuando el tablespace es vuelto a poner en online, oracle aplica el rollback a los datos en el tablespace si es necesario.

No se puede tener un tablespace offline si este contiene algún rollback segment que esté en uso. Cuando un tablespace esta en offline o regresa a online, este cambio se almacena en el diccionario de datos en el tablespace SYSTEM. Si un tablespace estuvo en estatus offline cuando se dio de baja la base de datos, este permanecerá en ese mismo estatus cuando la base de datos vuelva a ser montada o abierta.

Se puede tener un tablespace online solamente en la base de datos en el cual éste a sido creado por que la información necesaria está en el diccionario de datos y este a su vez esta contenido en el tablespace SYSTEM de esa base de datos. Oracle automáticamente cambia los tablespaces de online a offline cuando ha ocurrido algún error (por ejemplo, cuando el DBWR falla en diferentes intentos de escribir en un datafile del tablespace) . Los usuarios que intenten accesar a las tablas de ese tablespace con problemas reciben un mensaje de error. Si el problema lo causa el disco la falla es un "media failure", el tablespace debe ser recuperado después de que se corrija el problema de hardware.

Tablespaces de solo lectura

El propósito primordial de un tablespace de solo lectura es eliminar lo necesario para ejecutar un respaldo y recuperación de una porción grande de la base de datos. Oracle nunca actualiza los archivos de solo lectura del tablespace y de esta manera los archivos pueden residir en un medio de solo lectura tal como un CD ROM.

Siempre que se crea un tablespace nuevo, este es creado como solo de lectura, la opción READ ONLY del comando ALTER TABLESPACE permite hacer cambios en el tablespace para hacerlos de solo lectura o para que puedan ser modificados de nueva cuenta.

Tablespaces temporales

El espacio para manejo de operaciones de ordenamiento se ejecuta de manera más eficiente usando tablespaces temporales designados exclusivamente para esta tarea. Este esquema efectivamente elimina la serialización del espacio manejado para operaciones que están envueltas en la asignación y desasignación de este espacio. Todas las operaciones que usan ordenamientos incluyendo joins, indices y la cláusulas ORDER BY , GROUP BY y ANALIZE se ejecutan mejor desde tablespaces temporales.

Un tablespace temporal es un tablespace que solamente puede ser usada por segmentos de ordenamiento. Un segmento de ordenamiento existe en cualquier instancia que ejecute operaciones de ordenamiento dentro de un tablespace. Un tablespace temporal provee mejoras en el rendimiento cuando se usan múltiples ordenamientos que ocupan un gran tamaño en memoria.

Datafiles

Un tablespace en una base de datos oracle consiste en uno o más datafiles fisicos. Un datafile puede estar asociado con uno y solamente un tablespace y solamente una base de datos.
Cuando un datafile es creado para un tablespace, oracle crea el archivo asignando una cantidad especifica de espacio en disco más la cabecera requerida para el archivo de cabecera. Cuando un datafile es creado, el sistema operativo es responsable de limpiar la información vieja y autorizar antes de asignar esta para oracle. Si el archivo es grande, este proceso puede tardar una cantidad significativa de tiempo.

Siempre el primer tablespace en cualquier base de datos oracle es el tablespace SYSTEM, oracle automáticamente asigna los primeros datafiles de cualquier base de datos para el tablespace SYSTEM durante la creación de la base de datos.

Después que un datafile es creado, la asignación de espacio a disco no contiene aun datos, sin embargo, oracle reserva el espacio para mantener solamente los datos para futuros segmentos asociados al tablespace.

Se puede alterar el tamaño de un datafile después de su creación o se puede especificar que un datafile crezca dinámicamente como un objeto de un tablespace. Esta funcionalidad permite tener menos datafiles por tablespace y puede simplificar la administración de estos.

Datafiles Offline

Se pueden tener tablespaces offline (no disponibles) o tener algunos online (disponibles) al mismo tiempo, para esto, todos los datafiles creados para un tablespace tienen estatus offline como una unidad cuando el tablespace esta offline. Se pueden tener datafiles individuales en estatus de offline, sin embargo, normalmente se hace durante los procedimientos de recuperación.

OBJETOS DEL ESQUEMA

Para cada usuario de la base de datos está asociado un esquema (schema); el cual es una colección de objetos como tablas, vistas, secuencias, sinónimos, indices, clusters, ligas de base de datos, procedimientos y paquetes. Los objetos del esquema son estructuras lógicas de almacenamiento de datos y no cuentan con una correspondencia uno a uno con los archivos físicos en el disco dónde se almacena la información; sin embargo, oracle almacena los objetos del esquema dentro de un tablespace de la base de datos. Los datos de cada objeto están almacenados físicamente en uno o más datafiles de los tablespaces como se muestra el la **figura 18**.

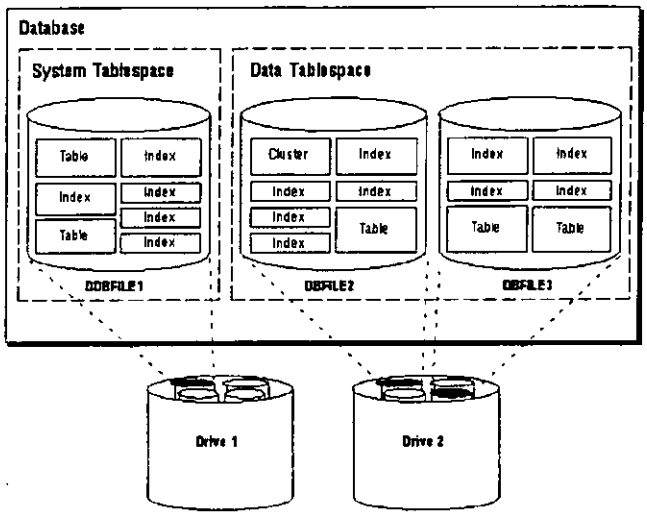


Fig. 18. Objetos del esquema, Tablespaces y Datafiles. Como podemos observar, el tablespace contiene objetos de diferentes esquemas y los objetos de un esquema pueden estar contenidos en tablespaces diferentes.

Tablas

Una tabla es una unidad básica de almacenamiento en oracle, los datos son almacenados en columnas y renglones y se define la tabla con un nombre (por ejemplo Empleados) y un conjunto de columnas también con un nombre y un tipo de datos (Varchar2, Number, Date, etc.) y un ancho de columna. Un renglón es una colección de columnas que contiene información de un solo registro.

Adicionalmente se pueden especificar reglas para cada columna de la tabla, estas reglas se pueden llamar constraints de integridad , ejemplo de esto es el NOT NULL, constraint que forza a que la columna contenga siempre un valor. Una vez creada la tabla se puede hacer sobre ella insert, delete, update o algunas consultas a través de querys usando SQL. En la **figura 19** podemos observar como está construida una tabla.

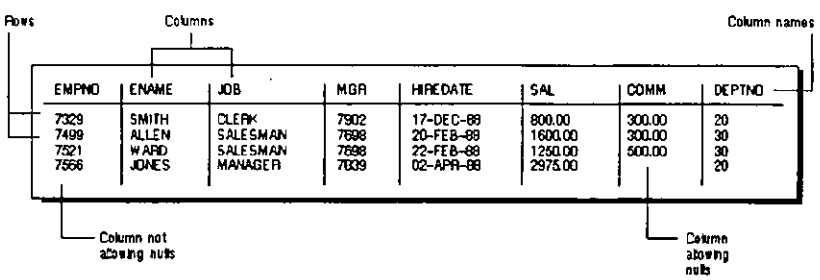


Fig. 19. Una tabla de oracle.

Vistas

Una vista es una presentación de los datos contenidos en una o más tablas o de otras vistas, contiene la salida de un query mostrándose como una tabla, de esta manera una vista puede ser tratada como un query almacenado o como una tabla virtual; por ejemplo, la tabla que se muestra en la figura 19 tiene diferentes columnas y un número de registros con información, si solo se necesita usar solamente cinco de esas columnas y solamente un conjunto de registros específicos se puede crear una vista para que otros usuarios la puedan accesar, podemos ver el ejemplo más claro en la **figura 20**.

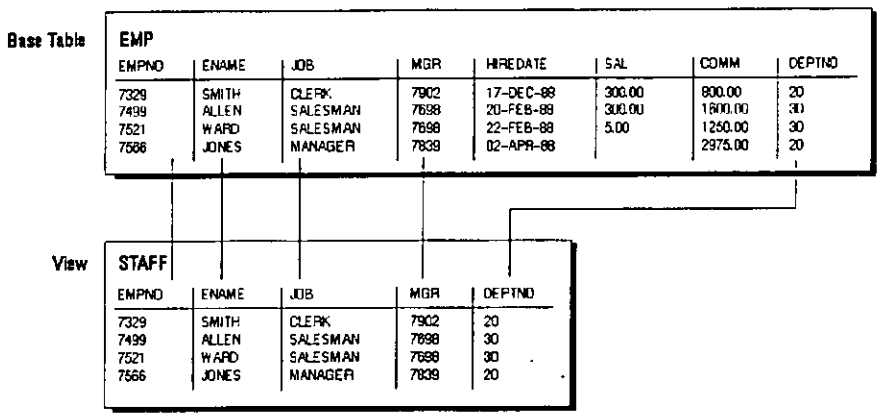


Fig. 20.

Las vistas son derivadas de las tablas y existen muchas similitudes entre las dos, por ejemplo, se pueden definir vistas hasta de 254 columnas de la misma manera que una tabla, se pueden hacer querys sobre las vistas y con las mismas restricciones que en las tablas, se puede insertar, actualizar y borrar desde las vistas y todas las operaciones sobre una vista afectaran los datos en la tabla base sobre la cual se creo la vista atendiendo a los constraint de integridad y triggers definidos en la tabla.

Generador de secuencias

El generador de secuencias provee una secuencia de números en serie, este es usado especialmente en entornos multiusuario para generar números de secuencia únicos, por consiguiente el generador de secuencias reduce la serialización cuando los comandos de dos transacciones puede generar el mismo número al mismo tiempo, además esto mejora también el tiempo de la transacción.

Los números de secuencia son enteros definidos en la base de datos de hasta 38 dígitos, una definición de secuencia indica en términos generales el nombre de la secuencia, si esta es ascendente o descendente, el intervalo entre los números y otra información. La parte más importante de la definición de la secuencia es cuando oracle debe almacenar los números generados por la secuencia en memoria. La definición de todas las secuencias de una base de datos oracle se almacenan como un registro en el diccionario de datos en el tablespace de SYSTEM, de esta manera, todas las secuencias siempre están disponibles por que el tablespace de system siempre está online.

Los números de secuencia son usados por comandos SQL que hacen referencia a la secuencia, se puede usar un comando para generar un nuevo número de secuencia o usar el número de secuencia actual. Solamente una sesión de usuario genera un número de secuencia y está disponible solo para esa sesión, cada usuario que hace referencia a la secuencia tiene acceso a su propio numero de secuencia actual.

Los números de secuencia son generados independientemente de las tablas, por consiguiente, el mismo generador de secuencias puede ser usado por una o múltiples tablas. La generación de una secuencia es usada para generar llaves primarias únicas automáticamente.

Sinónimos

Un sinónimo es un alias para una vista, tabla, snapshot, secuencia, procedimiento, función o paquete. Debido a que un sinónimo es un simple alias este requiere que no sea almacenado otro igual en el diccionario de datos, generalmente son usados estos por seguridad y conveniencia por ejemplo, proveen una mascara para el nombre de los objetos, transparencia para los objetos remotos de una base de datos distribuida y simplifican los comandos SQL para los usuarios de la base de datos.

Se pueden crear sinónimos públicos y privados, un sinónimo público es propiedad de un usuario especial llamado PUBLIC y todos los usuarios en la base de datos pueden accesarlo. Un sinónimo privado está contenido solo para un esquema especifico de usuario. Ambos tipos de sinónimo son usados para sistemas distribuidos y no distribuidos. Esto resulta ventajoso por que si el objeto puede ser movido o renombrado, solamente el sinónimo necesitaría ser redefinido y las aplicaciones basadas en el sinónimo continúan funcionando sin la modificación.

Clusters

Un cluster es un método opcional para almacenar datos de una tabla. Es un grupo de tablas que comparten los mismos data blocks debido a que comparten columnas comunes y comúnmente son usados juntos. Por ejemplo, si tenemos las tablas DEPT y EMP, las tablas compartes la columna DEPTNO; cuando se hace un cluster de estas tablas (figura 21), oracle almacena fisicamente los registros para cada departamento en las tablas EMP y DEPT dentro de los mismos datablocks.

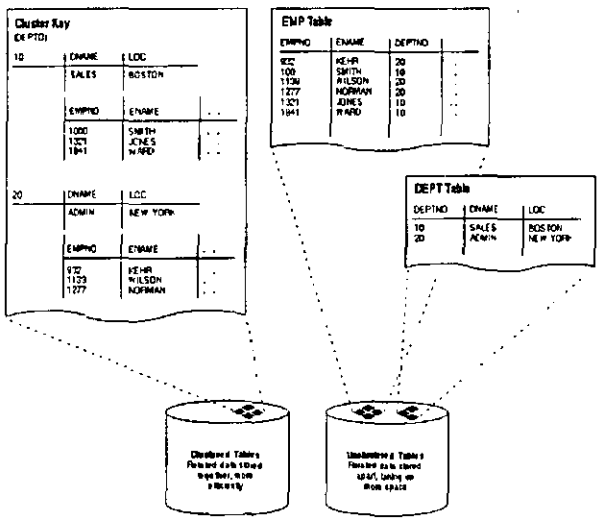


Fig. 21

Debido a que los clusters almacenan registros relacionados de diferentes tablas en los mismos data blocks usando clusters, se ofrecen estos dos beneficios:

- Las entradas y salidas del disco son reducidas en su tiempo de acceso y mejoran los joins para las tablas del cluster.
- En un cluster, el valor de una llave del cluster es también el valor de la llave de una columna para un registro en particular, cada valor de una llave del cluster es almacenado solamente una vez para cada cluster y su índice, de ninguna manera como muchos registros de diferentes tablas que contengan el valor.

De esta manera, se puede requerir menos espacio de almacenamiento para tablas relacionadas o índices en el cluster. Por ejemplo, cada llave del cluster (cada departamento) es almacenado solo una vez para muchos registros que contienen el mismo valor en ambas tablas (EMP y DEPT).

Hash Clusters

Este es un camino opcional para almacenar datos en una tabla y mejorar la ejecución en la recuperación de estos, para usar este método se puede crear un hash cluster y cargar tablas dentro del cluster. Oracle almacena físicamente los registros en el hash cluster y recupera estos de acuerdo a los resultado de una hash function, la cual se usa para generar una distribución de valores numéricos los cuales están basados en llaves especificas del cluster.

Para encontrar o almacenar un registro en una tabla indexada o en cluster, al menos dos accesos al disco son ejecutados (aunque generalmente son más), para mejorar esto se usa la hash function para localizar registros en el hash cluster y no se requieren accesos al disco, esto trae como resultado una operación mínima de I/O para leer o escribir en el hash cluster.

EJEMPLO DE CONFIGURACION DE LA BASE DE DATOS E INSTANCIA DE FIANZAS MONTERREY

En las siguientes páginas se muestra la configuración actual de la instancia y la base de datos oracle para Fianzas Monterrey, la idea es que partiendo de esta configuración en el capitulo referente a la afinación de la base de datos se pueda hacer una propuesta para mejorarlo dramáticamente.

Por otro lado, el fin de colocarlo al final de este primer capítulo es el de mostrar el uso de los conceptos que hasta aquí se han manejado.

```
Start
------------------------------------------------------------------------------
---Database Information as of : 12-Nov-2000, 09:09:43

                                About Database
                               ================

NAME      CREATED              LOG_MODE     CHECKPOINT_CHANGE# ARCHIVE_CHANGE#
--------- -------------------- ------------ ------------------ ---------------
FMAPROD   01/29/99 10:28:30    NOARCHIVELOG           24502240        24501501

                            About Oracle RDBMS Version
                                      =====

BANNER
----------------------------------------------------------------
Oracle7 Server Release 7.3.3.0.0 - Production Release
PL/SQL Release 2.3.3.0.0 - Production
CORE Version 3.5.3.1.0 - Production
TNS for HPUX: Version 2.3.3.0.0 - Production
NLSRTL Version 3.2.3.0.0 - Production



                                About LOGFILES
                               ================

                                                                        Size
Group Member                                                            (MB)
----- ---------------------------------------------------------- ----------
    4 /user/oracle/orabd/fmaprod/rdo041fmaprod.rdo                  30.0000
    5 /user/oracle/orabd/fmaprod/rdo051fmaprod.rdo                  30.0000
    6 /user/oracle/orabd/fmaprod/rdo061fmaprod.rdo                  30.0000

                               About CONTROL FILES
                               ================

ControlFile
----------------------------------------------------------------
/oracle/oradata/fmaprod/ctrl1fmaprod.ctl
/oracle/oradata/fmaprod/ctrl2fmaprod.ctl
```

/oracle/oradata/fmaprod/ctrl3fmaprod.ctl

About ROLLBACK SEGMENTS

Segment Name	Owner	Tablespace Name	Seg ID	Status	Size (MB)	EXTENTS
R01	SYS	RBS	14	ONLINE	10.1563	20
R02	SYS	RBS	15	ONLINE	10.1563	20
R03	SYS	RBS	16	ONLINE	10.1563	20
R04	SYS	RBS	17	ONLINE	10.1563	20
R05	SYS	RBS	18	ONLINE	10.1563	20
R06	SYS	RBS	19	ONLINE	10.1563	20
R07	SYS	RBS	20	ONLINE	10.1563	20
R08	SYS	RBS	25	ONLINE	10.1563	20
R09	SYS	RBS	21	ONLINE	10.1563	20
R10	SYS	RBS	22	ONLINE	40.6250	80
R11	SYS	RBS	23	ONLINE	10.1563	20
R12	SYS	RBS	24	ONLINE	20.8203	41
RBLONG	SYS	RBS	26	ONLINE	15.0000	3
SYSTEM	SYS	SYSTEM	0	ONLINE	.0977	2

About TABLESPACES

Tablespace Name	TS Size (In MB)	TS Free (In MB)	TS % Free
PRODCAT	200.00	169.23	84.61
PRODDAT	2300.00	1203.18	51.92
PRODIND	2200.00	838.49	40.06
RBS	300.00	121.99	40.66
SYSTEM	80.00	24.07	30.09
TOOLS	15.00	9.56	63.72
USERS	2.00	1.95	97.36
avg			58.35
sum	5097.00	2368.46	

About TABLESPACES

Tablespace Name	TS_SIZE	INITIAL_EXTENT	NEXT_EXTENT	PCT_INCREASE	Status
PRODCAT	200	10240	10240	0	ONLINE
PRODDAT	2300	10240	10240	0	ONLINE
PRODIND	2200	10240	10240	0	ONLINE
RBS	300	131072	131072	0	ONLINE
SYSTEM	80	10240	10240	50	ONLINE
TEMP	400	10240	10240	0	ONLINE
TOOLS	15	10240	10240	50	ONLINE
USERS	2	10240	4096	50	ONLINE

About TABLESPACES

TS_Name	TS_Size	TS_Free	(Free%)	Frag
PRODCAT	200	169.23	84.61	25
PRODDAT	700	300.15	42.88	267
PRODDAT	800	903.03	112.88	50
PRODIND	600	368.51	61.42	26
PRODIND	800	469.98	58.75	92
RBS	300	121.99	40.66	41
SYSTEM	80	24.07	30.09	12
TOOLS	15	9.56	63.72	13
USERS	1	1.95	194.73	2

avg		76.64	
sum	3496	2368.46	528

About Index Tablespace Use

Tablespace Name	Index_Count
PRODIND	411
SYSTEM	105
TOOLS	39
sum	555

About Data Files

File Name	Tablespace Name	Frag	Size	Status
/prod/oracle/orabd/fmaprod/dat1fmaprod.dbf	PRODCAT	25	200	AVAILABLE
/prod/oracle/orabd/fmaprod/dat2fmaprod.dbf	PRODDAT	25	800	AVAILABLE
/prod/oracle/orabd/fmaprod/dat3fmaprod.dbf	PRODDAT	25	800	AVAILABLE
/prod/oracle/orabd/fmaprod/dat4fmaprod.dbf	PRODDAT	267	700	AVAILABLE
/user/oracle/orabd/fmaprod/ind1fmaprod.dbf	PRODIND	48	800	AVAILABLE
/user/oracle/orabd/fmaprod/ind2fmaprod.dbf	PRODIND	44	800	AVAILABLE
/user/oracle/orabd/fmaprod/ind3fmaprod.dbf	PRODIND	26	600	AVAILABLE
/user/oracle/orabd/fmaprod/rbs1fmaprod.dbf	RBS	41	300	AVAILABLE
/oracle/oradata/fmaprod/sys1fmaprod.dbf	SYSTEM	12	80	AVAILABLE
/oracle/oradata/fmaprod/tools1prod.dbf	TOOLS	13	15	AVAILABLE
/oracle/oradata/fmaprod/usr1fmaprod.dbf	USERS	1	1	AVAILABLE
/prod/oracle/orabd/fmaprod/usr2fmaprod.dbf	USERS	1	1	AVAILABLE
sum		528		

About USERS

User Name	Default TS	Temporary TS	Created	Profile
AAISPURO	USERS	TEMP	08/08/2000	PROF_END_USER
AALONSO	USERS	TEMP	06/01/2000	PROF_END_USER
AANDRADE	USERS	TEMP	12/05/2000	PROF_END_USER
AAQUINO	USERS	TEMP	06/09/2000	DEFAULT
AARCO	USERS	TEMP	22/08/2000	DEFAULT
ABARRAGAN	USERS	TEMP	23/11/1999	PROF_END_USER
ABERMUDEZ	USERS	TEMP	22/08/2000	DEFAULT
ABONILLA	USERS	TEMP	22/08/2000	DEFAULT
AESQUINO	USERS	TEMP	22/08/2000	DEFAULT

-- Solo se incluyen los primeros nueve pero hacen un total de 122 usuarios creados.

About Parameter

ControlFile	VALUE
always_anti_join	NESTED_LOOPS
audit_trail	NONE
background_dump_dest	$ORACLE_HOME/dbs/
cache_size_threshold	2000
compatible	7.3.0
control_files	/oracle/oradata/fmaprod/ctrl1fmaprod.ctl, /oracle/oradata/fmaprod/ctrl2fmaprod.ctl, /oracle/oradata/fmaprod/ctrl3fmaprod.ctl

cpu_count	1
db_block_buffers	20000
db_block_lru_latches	1
db_block_size	2048
db_file_multiblock_read_count	32
db_name	fmaprod
distributed_transactions	54
dml_locks	500
enqueue_resources	520
gc_freelist_groups	50
gc_releasable_locks	20000
global_names	TRUE
job_queue_processes	1
log_buffer	655360
log_checkpoint_interval	10000
log_simultaneous_copies	0
max_dump_file_size	10240
mts_max_dispatchers	0
mts_max_servers	0
mts_servers	0
mts_service	fmaprod
nls_date_format	DD/MM/YYYY
open_cursors	300
optimizer_mode	CHOOSE
processes	175
remote_login_passwordfile	NONE
resource_limit	TRUE
rollback_segments	r01, r02, r03, r04, r05, r06, r07, r08, r09, r10, r11, r12, rblong
sequence_cache_entries	100
sequence_cache_hash_buckets	89
sessions	197
shared_pool_size	45000000
sort_area_retained_size	65536
sort_direct_writes	AUTO
temporary_table_locks	197
timed_statistics	TRUE
transactions	216
transactions_per_rollback_segment	16
user_dump_dest	$ORACLE_HOME/dbs/

About SGA

ControlFile	VALUE
Fixed Size	38980
Variable Size	50508080
Database Buffers	40960000
Redo Buffers	655360
sum	92162420

DataBase File IO Weights
Ordered By Drive

Drive	FILENAME	Total_IO	Weight
/orac	/oracle/oradata/fmaprod/sys1fmaprod.dbf	1327800	2.92
	/oracle/oradata/fmaprod/tools1prod.dbf	10274	.02
	/oracle/oradata/fmaprod/usr1fmaprod.dbf	6	.00
*****			
sum		1338080	2.94
/prod	/prod/oracle/orabd/fmaprod/dat1fmaprod.dbf	210164	.46
	/prod/oracle/orabd/fmaprod/dat2fmaprod.dbf	45470760	100.00

```
      /prod/oracle/orabd/fmaprod/dat3fmaprod.dbf          25606257   56.31
      /prod/oracle/orabd/fmaprod/dat4fmaprod.dbf          32224053   70.87
      /prod/oracle/orabd/fmaprod/tmp1fmaprod.dbf            577271    1.27
      /prod/oracle/orabd/fmaprod/usr2fmaprod.dbf                 0     .00
*****                                                  ---------- -------
sum                                                    104088505  228.91

/user /user/oracle/orabd/fmaprod/ind1fmaprod.dbf        8320985   18.30
      /user/oracle/orabd/fmaprod/ind2fmaprod.dbf        5816668   12.79
      /user/oracle/orabd/fmaprod/ind3fmaprod.dbf        2328538    5.12
      /user/oracle/orabd/fmaprod/rbs1fmaprod.dbf        1038776    2.28
*****                                                  ---------- -------
sum                                                     17504967   38.49

End
------------------------------------------------------------------------------
---Información de la base de datos al: 12-Nov-2000, 09:09:11
```

La base de datos como podemos observar contiene un gran número de objetos, algunos que están directamente relacionados a los usuarios como son las tablas, vistas, índices, etc. Por eso es necesario darle mantenimiento constante a esta para permitir ejecutar acciones sobre la base de datos y los objetos que pertenecen a esta con un gran nivel de seguridad. Así, los siguientes capítulos, marcan la pauta de las diferentes áreas en las que se tienen que concentrar para darle mantenimiento y así garantizar estos diferentes niveles de seguridad.

Capítulo II

Seguridad de la Base de Datos

La seguridad de la base de datos permite asignar o desasignar usuarios para ejecutar acciones sobre la base de datos y los objetos que pertenecen a esta. Oracle provee un discreto y comprensivo control de acceso. El control discrecional regula el acceso a todos los usuarios hacia los objetos de la base de datos a través de privilegios. Un privilegio es un permiso para accesar a un objeto dado de una manera preestablecida, por ejemplo, permisos para poder consultar una tabla. Debido a que los privilegios son dados a los usuarios en forma discreta por otros usuarios, esto es llamado seguridad discrecional. En la **figura 1** se muestran diferentes métodos de seguridad que el administrador de la base de datos puede usar.

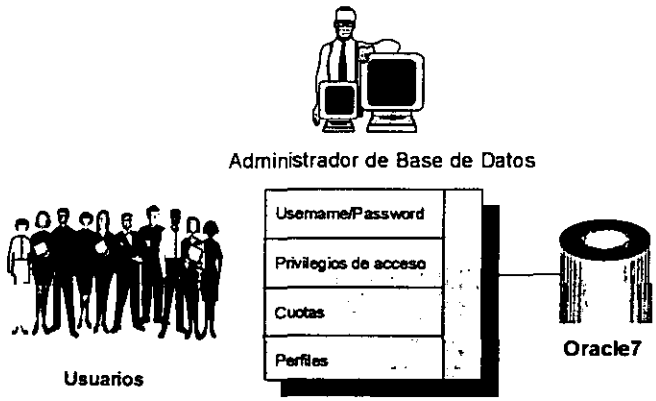


Fig. 1 Seguridad de la base de datos.

ESQUEMAS, USUARIOS DE LA BASE DE DATOS Y DOMINIOS DE SEGURIDAD

Los esquemas y usuarios ayudan al administrador de la base de datos a manejar la seguridad de la base de datos. Un esquema es un nombre para una colección de objetos tales como tablas, vistas, clusters, procedimientos y paquetes. Un usuario (algunas veces llamado como username) es un nombre definido en una base de datos que puede conectarse y accesar a los objetos del esquema de la base de datos.

Para accesar una base de datos un usuario debe correr una aplicación (tal como oracle forms o SQL plus) y conectarse usando un username definido en la base de datos.

Cuando un usuario de la base de datos es creado, un correspondiente esquema con el mismo nombre es creado también para ese usuario. Por default, un solo usuario se conecta a la base de datos, el usuario tiene acceso a todos los objetos contenidos en el esquema correspondiente. Un usuario es asociado solamente dentro del esquema del mismo nombre, sin embargo, el término de usuario y esquema es similar.

El acceso obliga al usuario a ser controlado a través de diferentes grupos de usuarios o dominios de seguridad de usuario. Cuando se crea un nuevo usuario en la base de datos o se altera uno existente, el administrador de seguridad debe tomar diferentes decisiones concernientes al dominio de seguridad de usuarios, esto incluye:

- Cuándo la información de autentificación de usuario debe ser mantenida por la base de datos. El sistema operativo o el sistema de autentificación de red.
- Grupos para los tablespaces temporales y de default del usuario.
- Una lista de los tablespaces accesibles para el usuario y las cuotas asociadas para cada tablespace listado.
- Los límites de recursos de usuario y el perfil; límites que dicten la cantidad de recursos del sistema disponibles para el usuario.
- Los privilegios y roles que proveen al usuario con accesos apropiados para accesar a los objetos que se necesitan para ejecutar las operaciones de la base de datos.

Todo lo anterior opera para todos los usuario excepto SYS y SYSTEM que debido a su dominio de seguridad estos nunca pueden ser alterados. En la **figura 2** se muestran los diferentes tópicos correspondientes a un dominio de seguridad.

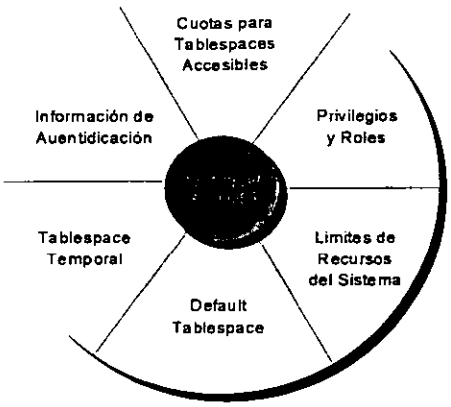


Fig. 2 Dominio de Seguridad

AUTENTIFICACIÓN DE USUARIOS

Para prevenir el uso de la base de datos a un username no autorizado, oracle provee una validación de usuario a través de tres diferentes métodos:

- Autentificación a través del sistema operativo
- Autentificación por una red de servicio de autentificación
- Autentificación a través de asociación con la base de datos Oracle.

Para simplificar, un método es usado generalmente para autentificar a todos los usuarios de la base de datos. Sin embargo, se permite usar todos los métodos dentro de la misma instancia de la base de datos. Oracle también encripta los passwords durante la transmisión para asegurar la autentificación del cliente/servidor.

Debido a que el administrador de la base de datos ejecuta operaciones especiales en la base de datos, oracle requiere procedimientos especiales de autentificación para el administrador.

Autentificación de Usuarios a Través del Sistema Operativo

Si el sistema operativo lo permite, oracle puede usar información del sistema operativo para autentificación de usuarios. Los beneficios de una autentificación de usuarios a través del sistema operativo son los siguientes:

- Los usuarios pueden conectarse a oracle más convenientemente (sin especificar un username o un password). Por ejemplo, un usuario puede hacer una llamada a SQL*Plus y saltar el username y password llegando directamente al prompt SQL>.
- El control sobre la autorización de usuarios esté centralizado en el sistema operativo. Oracle no necesita almacenar o manejar passwords. Sin embargo, oracle todavía mantiene usernames en la base de datos.
- Las entradas del usuario en la base de datos y el sistema operativo son auditadas de acuerdo a como corresponde normalmente.

Si el sistema operativo es usado para autentifiación de usuarios de la base de datos, existen algunas consideraciones especiales con respecto a un sistema de bases de datos distribuidas y links de bases de datos

Autentificación de Usuarios a través de la Red

Si los servicios de autentificación de red tales como DCE, Kerberos o SESAME están disponibles, oracle puede aceptar autentificación desde estos servicios. Para usar el servicio de autentificación de red con oracle, se debe también tener el producto Oracle Secure Network Services[1].

Si se usa el servicio de autentificación de red se debe tener especial consideración para los roles de red y los links de bases de datos.

Autentificación de Usuarios Usando la Base de Datos Oracle

Oracle puede autentificar usuarios cuando se intenta conectar a la base de datos usando información almacenada en esta base de datos. Se puede usar este método cuando el sistema operativo no puede ser usado por la base de datos para hacer la validación del usuario.

Cuando oracle usa la autentificación de la base de datos, se crea un password o contraseña asociado para cada usuario. Un usuario que provee un password correcto cuando se establece una conexión puede prevenir el uso de la base de datos a usuarios que no están autorizados. Oracle almacena el password del usuario en el diccionario de datos. Sin embargo, todos los passwords son almacenados en forma encriptada para mantener la seguridad del usuario. El usuario puede cambiar su password en cualquier momento.

Encriptación del Password Durante la Conexión

Para una mayor protección de la confidencialidad de los passwords, oracle permite encriptar passwords durante la conexión cliente/servidor o servidor/servidor. Si se habilita esta funcionalidad sobre el cliente y el servidor oracle podría usar la encriptación de passwords usando una modificación del algoritmo DES (Estándar de Encriptación de datos) antes de que sea enviado a la red.

Autentificación del DBA

El administrador de la base de datos a menudo debe ejecutar operaciones especiales tales como dar de baja la base de datos o iniciarla. Debido a esto, estas operaciones no pueden ser ejecutadas por usuarios comunes de la base de datos, los usernames de administrador necesitan un esquema más seguro de autentificación. Oracle provee algunos métodos para autentificación de administradores de bases de datos.

Dependiendo desde donde se desea administrar la base de datos, localmente sobre la misma máquina en la cual reside la base de datos o si se desea administrar muchas bases de datos diferentes de diferentes máquinas desde un cliente, se puede elegir entre autentificación por sistema operativo o archivos de password. En la **figura 3** podemos observar los esquemas de autentificación para un administrador de base de datos oracle.

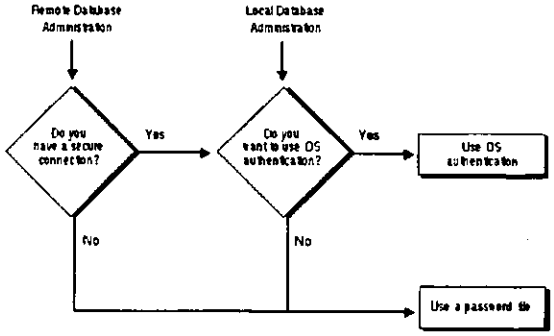


Fig. 3. Autentificación del DBA.

[1] Servicios de red para seguridad de oracle (Traducción)

En la mayoría de los sistemas operativos, la autentificación para el administrador de la base de datos usa el nombre de usuario del sistema operativo de un grupo especial (En UNIX por ejemplo este grupo es llamado el grupo DBA) o por algún proceso especial dado.

Los password files son archivos usados por la base de datos para guardar un registro de los usernames de la base de datos y de quien tiene privilegios de DBA (SYSDBA O SYSOPER). Estos privilegios permiten al administrador de la base de datos ejecutar las siguientes tareas:

SYSOPER Permite ejecutar STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, y Recover, incluyendo el privilegio de RESTRICTED SESSION.

SYSDBA contiene todos los privilegios de sistema con ADMIN OPTION y el privilegio de SYSOPER; permite CREATE DATABASE y recuperación de la base de datos.

Tablespaces de Usuarios

Como una parte del dominio de seguridad de usuarios, el administrador de la base de datos puede fijar varias opciones con respecto al uso de los tablespaces:

- El tablespace default de usuario
- Los tablespace temporales de usuario
- Uso de cuotas (espacios) sobre los tablespaces de la base de datos para el usuario

Default Tablespace

Cuando un usuario crea un objeto de esquema y no se especifica el tablespace en el que van a estar contenidos estos objetos, el objeto es puesto en el tablespace default de usuarios. Esto le permite a oracle el control del espacio usado en situaciones donde un objeto del tablespace no es especificado, el tablespace de usuario default es dónde este es creado; se puede cambiar después que el usuario ha sido creado.

Tablespace Temporal

Cuando un usuario ejecuta un comando SQL que requiere la creación de un segmento temporal, oracle asigna ese segmento en el tablespace temporal del usuario.

Acceso a Tablespace y Cuotas

Cada usuario puede ser asignado a una cuota de tablespace para algún tablespace de la base de datos. A través de la asignación de cuotas de tablespace de usuarios se logran dos cosas:

- El usuario puede usar el tablespace especificado para crear objetos, siempre que este tenga los privilegios apropiados.
- La cantidad de espacio que puede ser asignada para almacenar los objetos de los usuarios dentro del tablespace especificado puede ser limitada.

Por default, un usuario no tiene cuota sobre algún tablespace de la base de datos, por consiguiente, si el usuario tiene el privilegio para crear algún tipo de objeto de esquema, este deberá también ser asignado a una cuota del tablespace en la cual se creó el objeto o le será dado el privilegio para crear ese objeto en el esquema de otro usuario quien deberá tener asignada suficiente cuota de tablespace.

Se pueden asignar dos tipos de cuotas de tablespace para un usuario: una cuota para una cantidad especifica de espacio en el disco en el tablespace especificada en bytes, Kb o Mb o una cuota ilimitada[2].

Las cuotas de tablespace no son consideradas durante la creación de un segmento temporal:

- Los segmentos temporales no consumen alguna cuota que un usuario pueda poseer.
- Los segmentos temporales pueden ser creados en un tablespace para el cual un usuario no tenga una cuota.

Se puede asignar una cuota de tablespace para un usuario cuando se cree ese usuario y se puede cambiar esa cuota o agregar otra tiempo después.

El Usuario PUBLIC

Cada base de datos contiene un usuario de grupo llamado public, el usuario public provee un acceso público para objetos especificos del esquema (Tablas, vistas, etc.) proporcionado para todos los usuarios con privilegios especificos de sistema. Cada usuario automáticamente pertenece al grupo de usuarios de Public.

Como miembros de public, los usuarios pueden ver (select from) todos los datos de las tablas del diccionario de datos que tienen el prefijo USER, además, cualquier usuario puede dar permisos o crear un rol para public y todos los usuarios pueden usar los privilegios dados a public.

Se puede conceder o revocar cualquier privilegio de sistema, de objeto o de rol a public, sin embargo, para mantener la seguridad sobre los accesos, se deben otorgar solamente privilegios y roles para todos los usuarios para public.

Otorgar o quitar ciertos privilegios u objetos de sistema para public puede causar que alguna vista, procedimiento, función, paquete o trigger en la base de datos pueda ser recompilado.

Las restricciones para el usuario public son las siguientes:

- No se pueden asignar cuotas de tablespace para public, aunque se puede asignar el privilegio de sistema UNLIMITED TABLESPACE a public.
- Se pueden crear solamente links y sinónimos como objetos públicos usando CREATE PUBLIC DATABASE LINK/SYNONYM. Otros objetos no pueden ser propiedad de public por ejemplo el siguiente comando no es válido

CREATE TABLE PUBLIC.EMP...

Los segmentos de rollback pueden ser creados con la palabra clave public, pero este no es propiedad de public, todos los segmentos de rollback son propiedad de SYS.

USO DE LIMITES DE RECURSOS Y PERFILES

Como parte de la seguridad del dominio de usuario, se pueden tener límites sobre la cantidad de varios recursos del sistema disponibles para el usuario, explícitamente, para el límite de recursos para cada usuario, el administrador de seguridad puede prevenir el consumo incontrolado de recursos valiosos del sistema tales como el tiempo de CPU.

Las características del límite de recursos de oracle es muy usada por grandes sistemas multiusuario, algunos recursos de sistema son muy caros, por consiguiente el excesivo consumo de estos recursos por uno o más usuarios puede determinar y afectar a otros usuarios de la base de datos. Para un solo usuario o una pequeña escala de usuarios de un sistema de bases de datos multiusuario, las características de recursos del sistema no es útil por que los usuarios que consumen los recursos del sistema no causan un impacto perjudicial.

[2] Aunque en la práctica se deberían asignar cuotas especificas para prevenir que los objetos de usuario ocupen mucho espacio en un tablespace.

Se deben entonces usar los recursos de sistema a través de perfiles de usuario. Un perfil es un nombre de límites de recurso que se puede asignar a un usuario. Cada base de datos oracle puede tener un número ilimitado de perfiles. Además, oracle provee la opción de administrador de seguridad para habilitar o deshabilitar el perfil de recursos.

Si se usan los límites de recursos, ocurrirá una degradación en el rendimiento cuando los usuarios creen una sesión, esto debido a que los usuarios cargan todos los límites de recursos para ese usuario cuando se conecta a la base de datos.

Tipos de Recursos del Sistema y Límites.

Oracle puede limitar el uso de diferentes tipos de recursos del sistema. En general, se puede controlar cada uno de esos recursos a nivel de sesión en el nivel de llamado o en ambos.

Nivel de Sesión

Cada que el usuario se conecta a la base de datos, una sesión es creada, cada sesión consume tiempo de CPU y memoria en la computadora que ejecuta oracle. Diferentes límites de recursos pueden ser fijos para oracle en el nivel de sesión.

Si un usuario excede los recursos del nivel de sesión, oracle termina el comando actual (roll back) y retorna un mensaje indicando que el límite de sesión ha sido sobrepasado. En este punto, todos los comando previos en la transacción actual están intactos y solamente el usuario puede hacer un commit o roll back o desconectarse (en este caso la transacción actual es guardada); todas las operaciones producen un error. Después de que la transacción es guardada o cancelada, el usuario no puede trabajar más durante la sesión actual.

Nivel de Llamada

Cada que un comando SQL es ejecutado, se tienen que dar distintos pasos para ejecutar el proceso del comando. Durante este proceso, son hechas múltimples llamadas para la base de datos como parte de diferentes fases de la ejecución. Para prevenir alguna llamada de uso excesivo del sistema, oracle permite diversos límites de recursos para fijar el nivel de llamada.

Si un usuario excede el límites de recursos del nivel de llamada, oracle para el proceso del comando, hace un roll back y retorna un error. Sin embargo, todos los comandos previos de la transacción actual permanecen intactos y la sesión de usuario permanece conectada.

Tiempo de CPU

Cuando algún comando SQL y otros tipos de llamadas son hechas para oracle, es necesaria cierta cantidad de tiempo de CPU para procesar la llamada. En promedio, las llamadas requieren una pequeña cantidad de tiempo de CPU, empero, un comando SQL usa una gran cantidad de datos o bien un query puede potencialmente consumir grandes cantidades de tiempo de CPU, reduciendo el tiempo de CPU disponible para otros procesos.

Para prevenir el uso sin control de tiempo de CPU se puede limitar a este por llamada y la cantidad total de tiempo de CPU usado por llamadas de oracle durante la sesión. El límite es fijo y moderado a una centésima de segundo (0.01 segundos).

Lecturas Lógicas

La entrada y salida (I/O) es una de las operaciones más caras en un sistema de bases de datos. En una I/O intensa los comandos pueden monopolizar la memoria y disco usados y causar que otras operaciones de la base de datos tengan que competir por estos recursos.

Para prevenir las fuentes de I/O excesivas, oracle puede limitar las lecturas lógicas de los data blocks por llamadas y por sesión. Las lecturas de datablocks lógicas incluyen las lecturas desde los data blocks en la memoria y las lecturas desde el disco. El límite es moderado en el número de lecturas de bloques ejecutadas por una llamada o por sesión.

Otros Recursos

Oracle también provee para la limitación de diferentes recursos a nivel de sesión lo siguiente:

- Se puede limitar el número de sesiones concurrentes por usuario. Cada usuario puede crear solamente un número predefinido de sesiones concurrentes
- Se puede limitar el tiempo de ocio por sesión. Si el tiempo entre la llamada de oracle para una sesión alcanza el tiempo límite de ocio, la transacción actual es deshecha y la sesión actual es abortada, por consiguiente, los recursos de esa sesión son retornados al sistema. La siguiente llamada recibe un error que indica al usuario que no esta conectado a la instancia.

Poco tiempo después una sesión es abortada por que ha excedido el tiempo límite de ocio, PMON limpia después de haber sido abortada la sesión. Mientras PMON completa este proceso, la sesión muerta es contada aún como una sesión de sesión/usuario.

- Se puede limitar el tiempo de conexión por sesión. Si la duración de un inicio de sesión excede el tiempo límite, a la transacción actual se le hace Rollback y la sesión es eliminada, los recursos de la sesión son retornados al sistema.

Oracle no monitorea constantemente el tiempo de ocio o el tiempo de conexión, Esto ocasiona que el tiempo de ejecución del sistema se vea estropeado, en cambio, si esto se checa por lo menos cada cinco minutos, entonces oracle se vería forzado a abortar esa sesión.

- Se puede limitar la cantidad de espacio de SGA (usado para áreas de SQL privadas) por sesión. Este límite es solamente importante en sistemas que usan la configuración multitarea; de otra forma, las private SQL areas están alojadas en el PGA. Este límite es el número de bytes de memoria en el SGA de la instancia.

PERFILES

Un perfil es un conjunto de límites de recursos nombrado que puede ser asignado para validar usernames de una base de datos oracle. El perfil provee un fácil manejo de límites de recursos.

Solamente se necesita crear y manejar perfiles de usuario si los límites de recursos son un requerimiento para las políticas de seguridad la base de datos. Para usar perfiles, primero se categorizan los tipos de usuarios relacionados en la base de datos. Justamente como son usados los roles para manejar privilegios de usuarios relacionados los perfiles son usados para manejar los límites de recursos de los usuarios relacionados. Determinar qué tantos perfiles son necesarios para abarcar todos los tipos de usuarios en la base de datos y entonces determinar los apropiados límites de recursos para cada perfil.

Antes de crear los perfiles y fijar los límites de recursos asociados con estos, se deberían determinar los valores apropiados para cada límite de recurso, en base a estos valores sobre el tipo de operaciones que un usuario comúnmente ejecuta. Pro ejemplo, si un tipo de usuario no ejecuta normalmente un gran número de lecturas lógicas hacia un data block, entonces la variable LOGICAL_READS_PER_SESSION y LOGICAL_READS_PER_CALL pueden ser fijas en forma discreta.

Generalmente, el mejor camino para determinar los límites de recursos apropiado para un perfil de usuario dado es el de recoger información histórica acerca de cada tipo de recurso de usuario. Por ejemplo, el administrador de seguridad de la base de datos puede recoger información acerca de los límites de CONNECT_TIME, LOGICAL_READS_PER_SESSION y LOGICAL_READS_PER_CALL usando la característica de auditoria de oracle. A través del uso de la opción AUDIT_SESSION la auditoría genera información de ayuda que se puede usar para determinar los valores apropiados de los límites mencionados[3].

[3] Se habla aquí de los límites por perfil (profile). Se pueden generar también estadísticas para otros límites usando la caracterisitaca de monitor del server manager, específicamente las estadisitcas de monitor

Autorizaciones

Generalmente, oracle está autorizado para usar un máximo número de usuarios o para un máximo número de usuarios conectados en forma concurrente. El administrador de la base de datos es responsable de hacer seguro que el sitio cumpla con estos acuerdos para la licencia. La característica de autorización de oracle ayuda al administrador de la base de datos a que limite el número de sesiones conectadas concurrentemente para una instancia, o para limitar el número de usuarios creados en la base de datos y por eso asegurar que el sitio cumpla con los acuerdos de autorización o de licencia.

El administrador de base de datos controla los permisos y puede habilitar la facilidad de fijar los límites. Este puede también monitorear el uso del sistema. Si el administrador de la base de datos descubre que son necesarias más licencias o que son necesarias más sesiones puede solicitar una revisión de la versión a fin de contar con la cantidad de licencias apropiadas. Para hacer esto se tendria que contactar con el representante de oracle.

Uso de Licencias Concurrentes

En un uso concurrente de licencias, la licencia especifica un número de usuarios concurrentes, los cuales pueden estar conectados concurrentemente en la base de datos sobre una computadora específica en cualquier momento. Este número incluye todos los procesos batch y los usuarios en línea. También para un solo usuario con múltiples sesiones concurrentes, cada sesión cuenta por separado en el total del número de sesiones. Si un software permite la multiplexión (tal como TP monitor) es usado para reducir el número de sesiones directamente conectadas a la base de datos, el número de usuarios concurrentes es el numero de entradas distintas del multiplexor front-end.

El uso del mecanismo de licencias concurrentes permite al administrador de la base de datos lo siguiente:

- Un administrador puede fijar un límite sobre el número de sesiones concurrentes que puedan conectarse a una instancia a través del parámetro LICENCE_MAX_SESSIONS. Una vez agrandado este parámetro, solamente pueden conectarse usuarios que tengan el privilegio de sistema RESTRICTED_SESSION y puedan conectarse a la instancia; esto permite al administrador de la base de datos matar sesiones innecesarias permitiendo conectarse a otras sesiones.
- El administrador también puede fijar el límite de advertencias sobre el número de sesiones concurrentes que puedan conectarse a una instancia a través del parámetro LICENSE_SESSIONS_WARNING. Una vez agrandado este parámetro, oracle permite conectarse a sesiones adicionales pero envia un mensaje de advertencia a algún usuario con privilegio de RESTRICTED_SESSION en el ALTER file de la base de datos.

El administrador de la base de datos puede fijar esos límites en los parameter file de la base de datos de tal suerte que estos tengan efectos cuando la instancia se inicie. Alternativamente, el administrador puede cambiar estos mientras la instancia está corriendo usando el comando ALTER SYSTEM. Adicionalmente los mecanismos de licencias permiten al administrador de la base de datos checar el número de sesiones conectadas y el número máximo de sesiones concurrentes desde la instancia iniciada. El administrador puede usar esta información para evaluar las licencias del sistema necesarias y planear para actualizaciones de sistema.

El uso de límites concurrentes se aplica para todas las sesiones de usuario, incluyendo sesiones creadas por bases de datos remotas. Esto no se aplica para sesiones creadas por oracle o sesiones recursivas. Las sesiones que están conectadas a través de multiplexión externa. El administrador de la base de datos es responsable de hacer la cuenta de estas sesiones.

Uso de Licencias Nombradas

En el uso de licencias nombradas, la licencia especifica un número de usuarios, donde un usuario nombrado es un solo usuario que es autorizado para usar oracle sobre una computadora específica. No hay límites en el número de sesiones que cada usuario pueda tener concurrentemente o el número de sesiones concurrentes sobre la base de datos. Las licencias de usuario nombrado permiten al administrador de la base de datos fijar el límite sobre el número de usuarios que están definidos en la base de datos, incluyendo usuarios conectados vía link de base de datos. Este mecanismo asume que cada persona que accesa la base de datos tiene un nombre de usuario único en la base de datos y que no lo puede compartir con las demás personas.

Se puede fijar este limite en el parameter file de la base de datos para que surta efecto cuando la instancia es iniciada o bien hacerlo en linea con el comando ALTER SYSTEM.

PRIVILEGIOS

Un *privilegio* es un derecho de ejecutar un tipo particular de comando SQL o accesar a otros objetos de usuario, algunos ejemplos de privilegios son:

- Conectarse a la base de datos (crear una sesión)
- Crear una tabla
- Seleccionar registros desde una tabla de otro usuario
- Ejecutar procedimientos almacenados de otros usuarios.

Se otorgan privilegios a usuarios de modo que los usuarios pueden lograr la tarea requerida para realizar su trabajo. Se debe dar privilegios solamente al usuario quien absolutamente requiera el privilegio para realizar el trabajo necesario. La concesión excesiva de privilegios innecesarios pueden conducir a comprometer la seguridad. Un usuario puede recibir un privilegio de dos diferentes maneras:

- Se puede otorgar privilegios a usuarios explícitamente. Por ejemplo, se puede explícitamente otorgar el privilegio de insertar registros a la tabla EMP del usuario SCOTT.
- Se puede también otorgar privilegios a un rol (un grupo nombrado de privilegios), y entonces otorgar el rol a uno o más usuarios. Por ejemplo, se puede otorgar el privilegio de seleccionar, insertar, actualizar y borrar registros desde la tabla EMP a el rol llamado CLERK, el cual en su turno se puede otorgar a los usuarios SCOTT y BRIAN.

Debido a que los roles permiten un manejo mas fácil y mejor de privilegios, se deberia normalmente otorgar privilegios o roles a un usuario no especifico.

Existen dos categorias diferentes de privilegios:

- Privilegios de Sistema
- Privilegios de Objetos

Privilegios de Sistema

Un privilegio de sistema es el derecho de ejecutar una acción particular, o a ejecutar una acción particular en un *tipo* particular de objeto. Por ejemplo, el privilegio de crear Tablespaces y el de borrar un registro de cualquier tabla en una base de datos son privilegios de sistema. Existen sobre 60 privilegios de sistema diferentes

Se puede otorgar o revocar privilegios de sistema a usuarios y roles. Si los privilegios de sistema están otorgados a roles, la ventaja de los roles puede ser usada para manejar privilegios de sistema (por ejemplo, permisos de privilegios de roles a ser hechos selectivamente disponibles).
Los privilegios de sistema[4] se otorgan o revocan desde el usuario y roles usando también lo siguiente:

- El usuario o las carpetas de roles del server manager.
- Los comandos SQL GRANT Y REVOCKE.

[4] Usualmente se deberían otorgar privilegios de sistema solamente al personal administrativo y a los desarrolladores de aplicaciones por que el usuario final normalmente no requiere la capacidad asociada.

Unicamente usuarios con un rol especifico otorgado con el ADMIN OPTION o usuarios con el GRANT ANY PRIVILEGE (OTORGADOR DE CUALQUIER PRIVILEGIO). El administrador de base de datos o administradores de seguridad pueden otorgar o revocar privilegios de sistema a otros usuarios.

Estos privilegios son asignados a usuarios y roles con el comando GRANT o removidos con el comando REVOKE[5].

Privilegio de Sistema	*Explicación*
ANALYZE ANY	Analizar cualquier tabla, indice o cluster en cualquier esquema
AUDIT ANY	Auditar cualquier objeto en cualquier esquema
TRUNCATE ANY	Truncar cualquier tabla o cluster en cualquier esquema
CREATE CLUSTER	Crear un cluster en un esquema propio
ALTER ANY CLUSTER	Alterar cualquier cluster en cualquier esquema
CREATE ANY CLUSTER	Crear un cluster en cualquier esquema
DROP ANY CLUSTER	Eliminar cualquier cluster en cualquier esquema
ALTER DATABASE	Alterar la base de datos
CREATE DATABASE LINK	Crear links de base de datos privados en un esquema propio
CREATE PUBLIC DATABASE LINK	Crear links de base de datos públicos
CREATE INDEX	Crear un índice en el esquema propio o de cualquier tabla en el mismo esquema
ALTER ANY INDEX	Alterar cualquier índice en cualquier esquema
CREATE ANY INDEX	Crear un índice en cualquier esquema o cualquier tabla de cualquier esquema
DROP ANY INDEX	Borrar cualquier indice de cualquier esquema
CREATE PROCEDURE	Crear procedimientos almacenados, funciones, packages en el esquema propio
ALTER ANY PROCEDURE	Alterar procedimientos, funciones, etc en cualquier esquema
CREATE ANY PROCEDURE	Crear procedimientos, funciones, etc en cualquier esquema
DROP ANY PROCEDURE	Eliminar procedimientos, funciones, etc en cualquier esquema
EXECUTE ANY PROCEDURE	Ejecutar cualquier procedimiento almacenado o referenciar variables públicas
GRANT ANY PRIVILEGE	Otorgar privilegios de sistema aun cuando no sea el propietario
ALTER PROFILE	Alterar cualquier perfil en la base de datos
CREATE PROFILE	Crear perfiles
DROP PROFILE	Eliminar cualquier perfil en la base de datos
ALTER RESOURCE COST	Fijar costos por recursos de sesión
CREATE ROLE	Crear roles
ALTER ANY ROLE	Alterar cualquier rol en la base de datos
DROP ANY ROLE	Eliminar cualquier rol en la base de datos
GRANT ANY ROLE	Otorgar cualquier rol en la base de datos
ALTER ROLLBACK SEGMENT	Alterar segmentos de Rollback
CREATE ROLLBACK SEGMENT	Crear segmentos de Rollback
DROP ROLLBACK SEGMENT	Eliminar segmentos de Rollback
ALTER SESSION	Facilita el rastreo, características del lenguaje nacional y cierra un link
CREATE SESSION	Conectar a la base de datos
RESTRICTED SESSION	Logearse después que la base de datos se inicia usando STARTUP RESTRICT
CREATE SEQUENCE	Crear una secuencia en el esquema propio
ALTER ANY SEQUENCE	Alterar cualquier secuencia en cualquier esquema
CREATE ANY SEQUENCE	Crear una secuencia en cualquier esquema
DROP ANY SEQUENCE	Eliminar cualquier secuencia de cualquier esquema
SELECT ANY SEQUENCE	Referenciar cualquier secuencia en cualquier esquema
CREATE SNAPSHOT	Crear snapshot en el esquema propio, también se requiere CREATE TABLE
ALTER ANY SNAPSHOT	Alterar cualquier snapshot en cualquier esquema
CREATE ANY SNAPSHOT	Crear cualquier snapshot en cualquier esquema, también CREATE ANY TABLE
DROP ANY SNAPSHOT	Eliminar cualquier snapshot en cualquier esquema

[5] ORACLE7 Server SQL Language Reference. Part No 778-70. Pp 18-21

Privilegio de Sistema	*Explicación*
CREATE SYNONYM	Crear un sinónimo en el esquema propio
CREATE ANY SYNONYM	Crear un sinónimo en cualquier esquema
DROP ANY SYNONYM	Eliminar cualquier sinónimo en cualquier esquema, excepto sinónimos públicos
CREATE PUBLIC SYNONYM	Crear sinónimos públicos
DROP PUBLIC SYNONYM	Eliminar sinónimos públicos.
AUDIT SYSTEM	Audita los eventos del sistema
ALTER SYSTEM	Limita los recursos, procesos de servidor y dispatcher, cambia y archiva los redo log files, recuperación distribuda, checkpoint, verifica el acceso a los archivos.
CREATE TABLE	Crear tablas en el esquema propio, requiere UNLIMITED TABLESPACE o una cuota sobre un tablespace
ALTER ANY TABLE	Alterar cualquier tabla en cualquier esquema
BACKUP ANY TABLE	Respaldar cualquier tabla en cualquier esquema con la utilería export
COMMENT ANY TABLE	Comentar sobre cualquier tabla, vista o columna en cualquier esquema
CREATE ANY TABLE	Crear una tabla en cualquier esquema
DELETE ANY TABLE	Borrar registros desde cualquier taba, vista o snapshot en cualquier esquema
DROP ANY TABLE	Eliminar cualquier tabla en cualquier esquema
INSERT ANY TABLE	Insertar en cualquier tabla, vista o snapshot en cualquier esquema
LOCK ANY TABLE	Bloquear cualquier tabla en cualquier esquema
SELECT ANY TABLE	Consultar cualquier tabla, vista, snapshot en cualquier esquema
UPDATE ANY TABLE	Actualizar registros en cualquier tabla, vista o snapshot en cualquier esquema
ALTER TABLESPACE	Alterar tablespaces
CREATE TABLESPACE	Crear tablespaces
DROP TABLESPACE	Eliminar tablespaces
MANAGE TABLESAPCE	Poner en online u offlines a los tablespaces o para hacer backup
UNLIMITED TABLESPACE	Usar una cantidad ilimitada de cualquier tablespace
FORCE TRANSACTION	Forzar commit o rollback de una transacción propia en la base de datos local
FORCE ANY TRANSACTION	Forzar commit o rollback de cualquier transacción distribuida en la base de datos
CREATE TRIGGER	Crear triggers en el esquema propio
ALTER ANY TRIGGER	Habilitar, deshabilitar o compilar trigger en cualquier esquema
CREATE ANY TRIGGER	Crear un trigger en cualquier esquema asociado con cualquier tabla de cualquier esquema
ALTER USER	Alterar otros usuarios: password, métodos de autentificación, cuotas, tablespaces, asignaciones a perfiles y roles de default
BECOME USER	Usado por el import cuando se cargan datos dentro de un esquema de usuario
CREATE USER	Crear usuarios, asignar cuotas o cualquier tablespace, fijar el tablespace temporal y asignar perfiles
DROP USER	Borrar a otros usuarios
CREATE VIEW	Crear una vista en el esquema propio
CREATE ANY VIEW	Crear una vista en cualquier esquema
DROP ANY VIEW	Eliminar cualquier vista en cualquier esquema.

Privilegios de Objeto

Un privilegio de objeto es un privilegio o el derecho de ejecutar una acción en particular en una tabla especifica , vista, secuencia, procedimiento, función, o paquete. Por ejemplo, el privilegio de borrar registros desde la tabla DEPT es un privilegio de objeto. Dependiendo del tipo de objeto, existen diferentes privilegios de objetos. Algunos objetos del esquema (como los clusters, indexes, triggers, y database links) no tienen privilegios de objetos asociados , su uso es controlado con los privilegios de sistema. Por ejemplo, para alterar un cluster, un usuario debe ser propietario del cluster o tener el privilegio de sistema ALTER ANY CLUSTER.

Estos privilegios aplican para objetos especificos.

Privilegio de Objeto	*Explicación*
ALL	Todos los privilegios de objeto que pueden ser aplicados
ALL PRIVILEGES	Lo mismo que ALL
ALTER	Cambiar definiciones
DELETE	Borrar registros
EXECUTE	Ejecutar el objeto y referenciar variables
INDEX	Crear un indice sobre la tabla
INSERT	Insertar registros
REFERENCES	Crear un constraint que haga referencia a una tabla, no autorizado autorizado desde roles
SELECT	Consultar registros
UPDATE	Cambiar o modificar registros

En esta tabla se muestran los privilegios específicos que un objeto puede tener.

Privilegio de Objeto	*Tablas*	*Vistas*	*Secuencias*	*Procedures*	*Snapshots*
ALTER	X		X		
DELETE	X	X			
EXECUTE				X	
INDEX	X				
INSERT	X	X			
REFERENCES	X				
SELECT	X	X	X		X
UPDATE	X	X			

Los privilegios de objeto otorgados para una tabla, vista, secuencia, procedimiento, función o paquetes, se aplican en todo caso referenciando la base del objeto por nombre o usando un sinónimo. Por ejemplo, hay una tabla JWARD.EMP con un nombre sinónimo JWARD.EMPLOYEE. JWARD emita la siguiente declaración:

```
GRANT SELECT ON emp TO swilliams;
```

El usuario SWILLIAMS puede preguntar JWARD.EMP referenciando la tabla por nombre o usando el sinónimo JWARD.EMPLOYEE:

```
SELECT * FROM jward.emp;
SELECT * FROM jward.employee;
```

Si se otorga el privilegio en una tabla, vista, secuencia, procedimiento, función o paquetes a un sinónimo para el objeto, el efecto es el mismo como si no fuera usado un sinónimo. Por ejemplo, si JWARD quiso otorgar el privilegio SELECT para la tabla EMP a SWILLIAMS, JWARD pudo emitir también la siguiente declaración:

```
GRANT SELECT ON emp TO swilliams;
GRANT SELECT ON employee TO swilliams;
```

Si un sinónimo es mandado, todos los otorgamientos para los objetos subrayados permanecen en efecto, aun si los privilegios fueron otorgados por un sinónimo especifico.

Otorgando y Revocando Privilegios de Objeto

Los privilegios de objeto pueden ser otorgados o revocados desde un usuario o por roles, si se otorgan privilegios de objeto a roles, se pueden hacer los privilegios selectivamente disponibles. Los privilegios de objeto pueden ser otorgados a o revocados por usuarios y roles usando los comandos SQL GRANT y REVOKE respectivamente.

Un usuario automáticamente tiene todos los privilegios de objeto para el objeto contenido en el esquema que corresponde al nombre del usuario – en otras palabras, el esquema que el usuario tiene-. Un usuario puede otorgar cualquier privilegios de objeto sobre cualquier objeto que el o ella tengan a cualquier otro usuario o rol. Si el otorgamiento incluye la GRANT OPTION (del comando GRANT), el usuario puede otorgar el objeto privilegio a otro usuario; de otra forma , el usuario puede usar solamente el privilegio pero no otorgarlo a otro usuario .

Tópicos de Seguridad de Tablas

El privilegio de objeto para tablas permite seguridad en las tablas en dos diferente niveles:

Operaciones de Manipulación de Lenguaje y Datos. Los privilegios de DELETE, INSERT, SELECT, y UPDATE permiten respectivamente las operaciones de BORRAR, INSERTAR, SELECCIONAR, ACTUALIZAR DML, en una tabla o vista. Se debería otorgar estos privilegios a usuarios y roles que necesitan una vista o de manipular una tabla de datos.

Se pueden restringir los privilegios de INSERTAR y ACTUALIZAR para tablas o específicamente columnas de la tabla. Con un INSERT selectivo, un usuario con privilegios puede insertar un registro pero solo con valores para las columnas seleccionadas, todas las otras columnas reciben NULL o los valores faltantes de la columna. Con la fecha actualizada, un usuario puede actualizar solamente los valores de una columna especifica de un registro. La selección de los privilegios INSERT y UPDATE son usados para restringir el acceso de usuarios a datos sensitivos. Por ejemplo si no se desea que la entrada de datos de usuarios alteren la columna SAL de los empleados en la tabla, seleccione INSERT y/o UPDATE y los privilegios puede ser otorgados que excluyen la columna SAL. Alternativamente, una vista pudiera satisfacer esta necesidad para seguridad adicional.

Operación de Lenguaje de Definición de Datos

Los privilegios de ALTER, INDEX y REFERENCES permiten que operaciones DDL sean ejecutadas en una tabla. Porque estos privilegios permiten a otros usuarios alterar o crear dependencias en una tabla, se debiera otorgar el privilegio conservadoramente. Adicionalmente a estos privilegios, un usuario que intente ejecutar una operación DDL en una tabla puede necesitar otros privilegios de sistemas y/o de objetos (por ejemplo, al crear un trigger en una tabla, el usuario requiere el privilegio de objeto ALTER TABLE y el privilegio de sistema CREATE TRIGGER.

Como con los privilegios INSERTy UPDATE, el privilegio REFERENCES puede ser otorgado en una columna especifica de una tabla. El privilegio REFERENCES no dispone el otorgamiento a usar una tabla sobre la cual el otorgamiento es hecho como una llave primaria a cualquier llave foránea que otorgue el deseo de crear en su propia tabla. Esta acción es controlada con un privilegio especial porque la presencia de llaves foráneas restringe la manipulación de datos y la alteración de tablas puede ser hecha con una llave primaria. Para una columna especifica con privilegio de REFERENCES restringe el otorgamiento a usar la columna nombrada, lo cual, por supuesto, debe incluir al menos una llave primaria o única de la tabla maestra.

Tópicos de Seguridad con Vistas

El privilegio de objeto para vistas permite varias operaciones DML. Como en la ejecución de un comando DML sobre una vista, los privilegios de objeto DML para tablas pueden ser aplicados similarmente a vistas.

Privilegios Requeridos para Crear una Vista

Para crear una vista, se deben de conocer los siguientes requerimientos:

- Debe haber sido otorgado el privilegio de sistema CREATE VIEW (para crear una vista en su esquema) o CREATE ANY VIEW (Para crear una vista en el esquema de otro usuario), también explícitamente o vía un rol.
- Debe haber sido otorgados los privilegios de objeto SELECT, INSERT, UPDATE, y/o DELETE en todos los objetos de la base de datos subrayando la vista o los privilegios de sistema SELECT ANY TABLE INSERT ANY TABLE, UPDATE ANY TABLE y/o DELETE ANY TABLE. No se puede haber obtenido estos privilegios a través de roles.

Adicionalmente, si se intenta otorgar acceso a su vista a otros usuarios, se debe haber recibido el privilegio de los objeto base con la opción GRANT OPTION o al privilegio de sistema con la opción ADMIN OPTION. Si no se tienen permisos y se otorga acceso a la vista, es probable que no se pueda accesar la vista.

Incrementando Seguridad de las Tablas Usando Vistas. Al usar una vista, solo se requiere el privilegio apropiado para la vista en si misma. Las Vistas son útiles para adicionar dos o mas niveles de seguridad para tablas:

Una vista puede proveer acceso a la columna seleccionada de la tabla(s) base que define la vista. Por ejemplo, se puede definir una vista en la tabla EMP al mostrar solamente las columnas EMPNO, ENAME, y MGR:

```
CREATE VIEW emp_mgr AS
SELECT ename, empno, mgr FROM emp;
```

Una vista puede proveer valores-base de seguridad para la información en la tabla. Una cláusula WHERE en la definición de una vista que muestra solamente las registros seleccionadas en la tabla base asociada. Considerar los siguientes dos ejemplos:

```
CREATE VIEW lowsal AS
SELECT * FROM emp
WHERE sal < 10000;
```

La vista LOWSAL permite accesar a todos las registros de la tabla base EMP que tienen los valores salariales menores a 10000. La seguridad de valores-base en los valores salariales en un registro.

```
CREATE VIEW own_salary AS
SELECT ename, sal
FROM emp
WHERE ename = USER;
```

La vista OWN_SALARY usa la pseudocolumna USER. El valor en la pseudocolumna USER es siempre el usuario actual. En la vista OWN_SALARY, solamente las registros con un ENAME que concuerdan con el usuario que esta usando la vista son accesibles. La seguridad de valores-base están definidos en el acceso del usuario a la vista. Esta vista combina ambas, la seguridad columna-nivel y seguridad valor-base.

Tópicos de Seguridad en Procedimientos

El privilegio de objeto para procedimiento (incluyendo el procedimiento standalone, funciones y paquetería) es EXECUTE. Se debe otorgar estos privilegios solamente a usuarios que necesitan ejecutar un procedimiento.

Se puede usar procedimientos para adicionar un nivel de seguridad de base de datos. Un usuario requiere solamente los privilegios para ejecutar un procedimiento y no privilegios sobre los objetos subrayados que el código de acceso de los procedimientos. Escribiendo un procedimiento y otorgando solamente el privilegio EXECUTE un usuario (y no el privilegio en el

objeto referenciado por el procedimiento) puede estar forzado a accesar el objeto referenciado solamente a través del procedimiento.

Privilegios Necesarios para Crear o Alterar un Procedimiento

Para crear un procedimiento, un usuario debe tener el privilegio de sistema CREATE PROCEDURE o CREATE ANY PROCEDURE. Para alterar un procedimeitno que ha sido compilado manualmente, el usuario debe ser su propietario o tener el privilegio de ALTER ANY PROCEDURE.

Adicionalmente, el usuario que es propietario del procedimiento debe tener el privilegio requerido para el objeto referenciado en el cuerpo del procedimiento. Para crear un procedimiento, se deben haber otorgado los privilegios necesarios (de sistema y/o de objetos) sobre todos los objetos referenciados por el procedimiento almacenado; no se puede haber obtenido el privilegio requerido a través de roles. Esto incluye el privilegio EXECUTE para cualquier procedimiento que es llamado dentro de un procedimiento almacenado que ha sido creado. Los triggers también requieren privilegios dados explícitamente por el dueño del trigger. Un bloque anónimo de PL/SQL puede usar cualquier privilegio siempre y cuando el privilegio sea dado explícitamente o vía rol.

Dominios de Seguridad y ejecución de Procedimientos

Un usuario con privilegio de EXECUTE para un procedimiento especifico puede ejecutar el procedimiento. Un usuario con el privilegio de sistema de EXECUTE ANY PROCEDURE puede ejecutar cualquier procedimiento en la base de datos. A un usuario se le pueden dar privilegios para ejecutar procedimientos a través de roles.

Cuando se ejecuta un procedimiento este opera bajo un dominio de seguridad de un usuario quien es el propietario del procedimiento. Sin embargo, puede no necesitar privilegios para el objeto. Debido a que el propietario de un procedimiento debe tener los privilegios de objeto necesarios, tendrían que darse menos privilegios para usuarios del procedimiento y así tener un mejor control del acceso a la base de datos.

Los privilegios actuales de el propietario de un procedimiento almacenado siempre son checados antes de que el procedimiento sea ejecutado. Si un privilegio necesario sobre el objeto es revocado para el propietario del procedimiento, este no podrá ser ejecutado ni por el propietario ni por algún otro usuario.

La ejecución de un trigger sigue estas mismas características. El usuario ejecuta un comando SQL y este tiene privilegios para ejecutarlo. Como resultado del comando SQL un trigger es disparado. El comando dentro de la acción temporal del trigger se ejecuta sobre un dominio de seguridad de el usuario que es propietario del trigger.

Paquetes y Objetos de los Paquetes

Un usuario con el privilegio de EXECUTE para un paquete puede ejecutar cualquier procedimiento público o función en el paquete y accesar o modificar el valor de alguna variable pública del paquete. El privilegio de execute no puede ser dado para una construcción de un paquete, debido a esto, se pueden considerar dos alternativas para establecer seguridad cuando se desarrollen procedimientos, funciones y paquetes para una aplicación de base de datos. Estas alternativas son descritas en los siguientes ejemplos:

EJEMPLO 1

Este ejemplo muestra cuatro procedimientos creados en los cuerpos de dos paquetes.

```
CREATE PACKAGE BODY hire_fire AS
      PROCEDURE hire(...) IS
            BEGIN
                  INSERT INTO emp . . .
            END hire;
      PROCEDURE fire(...) IS
```

```
            BEGIN
                  DELETE FROM emp . . .
            END fire;
END hire_fire;

CREATE PACKAGE BODY raise_bonus AS
      PROCEDURE give_raise(...) IS
            BEGIN
                  UPDATE EMP SET sal = . . .
            END give_raise;
      PROCEDURE give_bonus(...) IS
            BEGIN
                  UPDATE EMP SET bonus = . . .
            END give_bonus;
END raise_bonus;
```

El acceso para ejecutar el procedimiento es dado por el privilegio de EXECUTE para el paquete como en los siguientes comandos:

```
GRANT EXECUTE ON hire_fire TO big_bosses;
GRANT EXECUTE ON raise_bonus TO little_bosses;
```

El método de seguridad para los objetos del paquete no es discriminatorio para algún objeto en el paquete. El privilegio de EXECUTE dado para el paquete permite accesar a todos los objetos de este.

EJEMPLO 2

Este ejemplo muestra cuatro definiciones de procedimientos dentro del cuerpo de un solo paquete. Dos procedimientos standalone adicionales y un paquete son creados especificamente para proveer acceso a los procedimientos definidos en el paquete principal.

```
CREATE PACKAGE BODY employee_changes AS
      PROCEDURE change_salary(...) IS BEGIN ... END;
      PROCEDURE change_bonus(...) IS BEGIN ... END;
      PROCEDURE insert_employee(...) IS BEGIN ... END;
      PROCEDURE delete_employee(...) IS BEGIN ... END;
END employee_change;

CREATE PROCEDURE hire
      BEGIN
            insert_employee(...)
      END hire;

CREATE PROCEDURE fire
      BEGIN
            delete_employee(...)
      END fire;

PACKAGE raise_bonus IS
      PROCEDURE give_raise(...) AS
            BEGIN
                  change_salary(...)
            END give_raise;

      PROCEDURE give_bonus(...)
            BEGIN
                  change_bonus(...)
            END give_bonus;
```

Usando este método, los procedimientos que actualmente están trabajando (Los procedimientos en el paquete EMPLOYEE_CHANGES) son definidos en un paquete y son compartidas variables globales y cursores. Al declarar el nivel máximo de los procedimientos HIRE y FIRE y el paquete adicional RAISE_BONUS, se puede indirectamente dar permisos de EXECUTE sobre los procedimientos del paquete principal.

```
GRANT EXECUTE ON hire, fire TO big_bosses;
GRANT EXECUTE ON raise_bonus TO little_bosses;
```

ROLES

Oracle provee para el fácil manejo y control de privilegios el manejo a través de roles. Los roles son grupos nombrados de privilegios relacionados que se otorgan a los usuarios o a otros roles. Los roles están diseñados para facilitar la administración de los usuarios finales a través de privilegios de objeto y de sistema. Sin embargo, los roles no deben ser usados por los desarrolladores de una aplicación debido a que los privilegios para accesar a los objetos dentro de la programación almacenada necesita ser otorgada directamente.

Estas propiedades de los roles permiten facilitar el manejo de privilegios dentro de la base de datos:

- Reduce la administración de privilegios. en lugar de dar el mismo grupo de privilegios a diferentes usuarios se puede dar el mismo privilegio a un grupo de usuarios relacionados al rol, entonces, el solamente se necesita dar el privilegio al rol y no a cada miembro del grupo.
- Manejo dinámico de privilegios. Si los privilegios para un grupo deben cambiar, solamente los privilegios del rol necesitan modificarse. Los dominios de seguridad de todos los usuarios con privilegios del rol, automáticamente reflejarán los cambios hechos para el rol.
- Disponibilidad selectiva de privilegios. Se puede habilitar o deshabilitar selectivamente los roles dados a los usuarios. Esto permite un control especifico de los usuarios con privilegios en alguna situación dada.
- Conocimiento de la aplicación. Debido a que el diccionario de datos almacena cuales roles son los que existen, se puede diseñar la base de datos de la aplicación para consultar el diccionario de datos y automáticamente habilitar o deshabilitar roles selectivos cuando un usuario intente ejecutar una aplicación a través de un nombre de usuario.
- Seguridad de aplicaciones especificas. Se puede proteger un rol usando un password. Las aplicaciones pueden ser creadas especificamente para habilitar un rol cuando se suministre el password correcto. Los usuarios no pueden habilitar el rol si este no conoce el pasword.

Uso común de Roles

En general se crea un rol para servir a uno o más propósitos: el manejo de privilegios para la base de datos de la aplicación o para el manejo de privilegios de un grupo de usuarios. En la **figura 4** se muestra el uso de los roles.

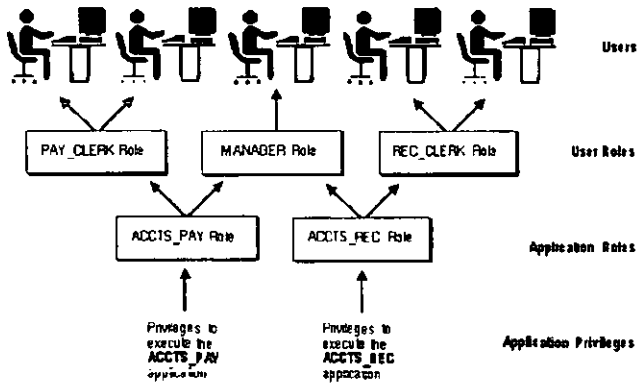


Fig. 4 Uso de Roles

Roles de aplicación

Se autoriza a un rol de aplicación todos los privilegios necesarios para correr una aplicación dada de base de datos. Entonces, se puede dar privilegio a de ese rol a otros roles o a usuarios específicos. Una aplicación puede tener diferentes roles con cada rol asignado a diferentes grupos de privilegios que permitan accesar a una mayor o menor cantidad de datos mientras se esté usando la aplicación.

Roles de usuario

Se puede crear un rol para un grupo de usuarios de la base de datos con requerimientos comunes de privilegios. Se manejan los privilegios de usuario a través de autorizar roles y privilegios al rol de usuario.

Mecanismo de los roles

La funcionalidad de los roles de base de datos incluye lo siguiente:

- Un rol puede permitir un privilegio de sistema o de objeto
- Un rol puede permitir otros roles. Sin embargo, un rol no puede autorizarse a si mismo y no puede autorizarse circularmente (Por ejemplo, el rol A no puede ser dado al rol B si el rol B ha sido previamente autorizado por el rol A)
- Cualquier rol puede ser dado para cualquier usuario de la base de datos.
- Cada rol autorizado para un usuario es en un tiempo dado habilitado o deshabilitado. El dominio de seguridad de usuarios incluye los privilegios de todos los roles concurrentemente habilitados para el usuario. Un dominio de seguridad de usuarios no incluye los privilegios de algunos roles concurrentemente deshabilitados para el usuario. Oracle permite a las aplicaciones de la base de datos y usuarios, habilitar y deshabilitar roles para proveer una disponibilidad selectiva de privilegios.
- Una autorización indirecta a un rol (un rol que autoriza a otro rol) puede ser explícitamente habilitado o deshabilitado para un usuario. Sin embargo, habilitando a un rol que contiene otro rol, implícitamente se habilitan todas las autorizaciones indirectas a roles.

Autorizando y Revocando Roles

Se puede autorizar o revocar roles desde usuarios u otros roles usando las siguientes opciones:

- Con el server manager agregando o removiendo privilegios desde una caja de diálogo.[6]
- Con los comandos SQL GRANT y REVOKE

Los privilegios son autorizados o revocados desde los roles usando la misma opción. Los roles pueden ser autorizados o revocados desde los usuarios usando el sistema operativo que ejecuta oracle.

Algunos usuarios con el privilegio de sistema GRANT ANY ROLE pueden autorizar o revocar cualquier rol para otros usuarios o roles de la base de datos. Se debe autorizar con mucha cautela este privilegio de sistema debido a que es muy poderoso. Además, cualquier usuario autorizado a un rol con la opción ADMIN OPTION puede autorizar o revocar ese rol para otro usuario o rol de la base de datos. Esta opción permite poder administrativo para los roles sobre una base selectiva.

Dentro de una base de datos, cada nombre de rol debe ser único y un nombre de rol y de usuario no pueden ser iguales. Los objetos del esquema y los roles no pueden estar contenidos en cualquier esquema, debido a esto, un usuario que crea un rol puede ser eliminado sin efectos sobre el rol.

Cada rol y usuario tienen su propio y único dominio de seguridad. Un dominio de seguridad de rol incluye privilegios autorizados par el rol más los privilegios otorgados por otros roles. Un dominio de seguridad de usuario incluye privilegios sobre todos los objetos en el correspondiente esquema, los privilegios dados para el usuario y los privilegios de roles otorgados para el usuario

[6] El server manager es una herramienta que como muchas otras permite la administración de la base de datos oracle.

que esta actualmente habilitado. Un dominio de seguridad de usuario incluye los privilegios y roles dados para el usuario PUBLIC. Un rol puede ser simultáneamente habilitado por un usuario y deshabilitado por otro.

Comandos de Lenguaje de Definición de Datos y Roles

Dependiendo del comando, un usuario requiere uno o más privilegios para ejecutar exitosamente un comando DDL[7]. Por ejemplo, para crear una tabla. El usuario debe tener el privilegio de sistema CREATE TABLE o CREATE ANY TABLE. Para crear una vista de una tabla de otro usuario, el creador requiere el privilegio de sistema CREATE VIEW o CREATE ANY VIEW y el privilegio de SELECT sobre la tabla o el privilegio SELECT ANY TABLE.

Oracle evita la dependencia sobre privilegios recibidos vía roles a través de restringir el uso de privilegios específicos en ciertos comandos DDL. Las siguientes reglas están dadas para estos casos:

- Todos los privilegios de sistema y privilegios de objeto que permitan al usuario ejecutar operaciones DDL son utilizables cuando son recibidos vía rol. A excepción del privilegio de objeto REFERENCES para una tabla que no puede ser usado para definición de una llave foránea de una tabla si el privilegio es recibido vía rol.
- Todos los privilegios de sistema y privilegios de objeto que permitan ejecutar operaciones DML[8] que es requerida para usar un comando DDL no es utilizable cuando se recibe vía rol. Por ejemplo, si un usuario recibe el privilegio de sistema SELECT ANY TABLE o el privilegio de objeto SELECT para una tabla vía rol, no se puede usar algún privilegio para crear una vista de alguna tabla de otro usuario.

El siguiente ejemplo muestra claramente los permisos y restricciones usados por privilegios recibidos vía roles:

Asumiremos que un usuario:

- Tiene autorizado un rol que tiene el privilegio de sistema CREATE VIEW
- Tiene autorizado un rol que tiene el privilegio SELECT para la tabla EMP
- No tiene directamente autorizado el privilegio de SELECT para la tabla EMP
- Tiene autorizado el privilegio de objeto SELECT para la tabla DEPT

Dados directa o indirectamente estos privilegios entonces:

- El usuario puede utilizar comandos SELECT sobre las tablas EMP o DEPT
- Aunque el usuario tiene tanto el privilegio de CREATE VIEW y el privilegio SELECT sobre la tabla EMP (ambos vía rol), el usuario no puede crear una vista sobre esta tabla por que el privilegio SELECT para la tabla EMP fue dado vía rol.
- El usuario puede crear una vista sobre la tabla DEPT debido a que este tiene el privilegio CREATE VIEW (vía rol) y el privilegio de SELECT sobre la tabla DEPT(dado directamente)

Los roles CONNECT, RESOURCE, DBA, EXP_FULL_DATABASE y IMP_FULL_DATABASE son definidos automáticamente por la base de datos oracle.

En algunos entornos se puede administrar la seguridad de la base de datos usando el sistema operativo. EL sistema operativo puede ser usado para manejar las autorizaciones (y para revocar estas) de roles de base de datos y/o manejar passwords de autentificación.

Otra forma de garanitzar la seguridad en los datos son las tareas de respaldo y recuperación que en el siguiente capitulo se detallan, estas son de un nivel superior y garantizan la integridad de los datos de la base de datos oracle por si se diera el caso de una pérdidad total.

[7] Lenguaje de Definicion de Datos, usado para crear objetos de la base de datos oracle.
[8] Lenguaje de Manipulación de Datos Permite insertar, borrar, consultar y actualizar datos en oracle

Capítulo III

Respaldo y Recuperación de la Base de Datos

El proceso de recuperación

Una de las responsabilidades del administrador de la base de datos es prepararse para posibles fallas en el sistema ya sean de hardware, software, red o procesos. Si estas fallas afectan la operación de la base de datos, se debe recuperar y retornar a la operación normal lo más rápidamente posible. La recuperación debería proteger la base de datos y los usuarios asociados para evitar problemas innecesarios y reducir las posibilidades de tener que duplicar el trabajo manualmente.

El proceso de recuperación varia dependiendo del tipo de falla que ha ocurrido, las estructuras que afectó y el tipo de recuperación que se ejecutará. Si se perdieron o se dañaron algunos archivos, la recuperación puede aumentar para no reiniciar una instancia, si los datos se han perdido, la recuperación requerirá pasos adicionales.

Diferentes tipos de problemas pueden parar la operación normal de la base de datos oracle o afectar el acceso de entrada y salida del disco. Para algunos de estos problemas la recuperación es automática y requieren una pequeña o ninguna acción de parte del administrador de la base de datos.

Errores de Usuario

Un administrador de base de datos puede prevenir los pequeños errores de usuario (por ejemplo, borrados accidentales de tablas). Generalmente los errores de usuario pueden ser reducidos incrementando el aprendizaje de las aplicaciones principales de la base de datos, además, planeando una efectiva recuperación adelantada del esquema, el administrador puede facilitar el trabajo necesario para recuperar desde muchos tipos de errores de usuario.

Errores de Comando

Los errores de comando ocurren cuando se da un error lógico en el manejo de los comandos en un programa de oracle. Por ejemplo, asumir que todos los extents de una tabla (El número de extents especificados en el parámetro MAXEXTENTS del comando CREATE TABLE) estén asignados y están completamente llenos con datos; La tabla está absolutamente llena. Un comando INSERT válido no puede insertar un registro debido a que no hay espacio disponible; por consiguiente, el comando falla.

El proceso background de oracle PMON detecta el proceso abortado, si este es un proceso de usuario o de servidor, PMON resuelve el problema de la falla a través de un roll back de la transacción actual o del proceso abortado y libera el o los recursos que este proceso estuvo usando. La recuperación de la falla de usuario o de servidor es automática. Si el proceso abortado es un proceso background, la instancia no puede continuar su operación correcta (generalmente). Para esto hay que desmontar y reiniciar la instancia.

Fallas de red

Cuando un sistema usa redes (Por ejemplo LAN, líneas telefónicas, etc.) para conectar una estación de trabajo cliente a una base de datos, la red puede fallar (tal como fallas en la conexión del teléfono o fallas en la comunicación por la red, etc.) puede interrumpir la operación normal de al base de datos. Por ejemplo: Una falla en la red podría interrumpir la ejecución normal de una aplicación del cliente y causar que ocurran fallas. En este caso, el proceso background PMON detecta y resuelve el proceso abortado.

Falla de la Instancia de la Base de Datos

Una falla de la instancia de la base de datos ocurre cuando un problema evita que se continúe trabajando. Esto puede ser resultado de un problema de hardware o de software. La recuperación de la instancia es relativamente automática.

Fallas de disco

Un error puede generarse cuando se intenta leer o escribir un archivo que es requerido para operar una base de datos oracle. Esta ocurrencia es llamada **"media failure"** por que este es un problema físico de lectura o escritura en los archivos físicos necesarios para la operación normal de la base de datos.

Un ejemplo común de un media failure es un disco que ha fallado, lo cual causa la pérdida de todos los archivos sobre un disco. Todos los archivos asociados con la base de datos son vulnerables a una falla en disco, incluyendo data files, redo log files y control files. Una recuperación apropiada de un media failure depende de los archivos afectados.

Los errores de disco pueden afectar uno o todos los tipos de archivos necesarios para la operación de la base de datos oracle, incluyendo datafiles, online redo log files y control files.

La operación de la base de datos después de un error de disco de un redo log file en línea o un control file depende si el redo log file online o el control file está multiplexado. Un redo log file o un control file multiplexado significa que existe una segunda copia de los archivos. Si un media failure daña un solo disco y se tiene un redo log online, la base de datos puede continuar con su operación sin una interrupción significativa. El daño a un redo log no multiplexado causa que la operación de la base de datos se detenga y puede causar una pérdida de datos permanente; el daño a cualquier control file si es multiplexado o no, detiene la operación de la base de datos una vez que oracle intenta leer o escribir en el control file dañado.

Los errores que afectan datafiles no pueden leer un datafile, entonces es retornado un error por el sistema operativo hacia la aplicación indicando que el archivo no pudo ser encontrado, no pudo ser abierto o no pudo ser leído. Oracle continua corriendo pero el error continua ocurriendo cada momento.

ESTRUCTURAS USADAS PARA LA RECUPERACIÓN DE BASE DE DATOS

Respaldo de la Base de Datos

Un respaldo de la base de datos consiste en hacer un respaldo a través del sistema operativo de los archivos físicos que constituyen una base de datos oracle. Para comenzar una recuperación de la base de datos después de un media failure, oracle usa archivos respaldados para restaurar los datafiles dañados y los control files.

El redo log

El redo log, presente en toda base de datos oracle, registra todos los cambios hechos en una base de datos. El redo log de una base de datos consiste en más de dos redo log files que están separados de los data files (el cual almacena actualmente los datos de la base de datos). Como parte de una recuperación de la base de datos de una instancia o media failure, oracle aplica los cambios apropiados en los redo log files de la base de datos los cuales actualizan los datos de la base de datos en el instante que la falla ocurrió.

Un redo log de la base de datos puede estar comprendido en dos partes: el redo log file on line y el archived redo log file.

El redo log file online. Toda base de tiene un redo log file online asociado. El redo log file online trabaja con el proceso background LGWR para almacenar inmediatamente todos los cambios hechos a través de la instancia asociada. El redo log file online consiste en dos o más archivos preasignados que son rehusados en forma circular para continuar almacenando los cambios de la base de datos.

El archived (offline) Redo log. Opcionalmente se puede configurar la base de datos oracle para guardar archivos de redo log files una vez que estos se llenan. El redo log file online es archivado para una más extensiva operación de recuperación.

Segmentos de Roll Back

Los segmentos de rollback son usados por un número significativo de funciones en la operación de la base de datos oracle. En general, los segmentos de rollback de una base de datos almacenan los valores viejos de los datos cambiados por las transacciones. Entre otras cosas, la información en un segmento de rollback es usada durante la recuperación para deshacer cualquier cambio aplicado desde el redo log para los data files.

Control Files

En general, el control file de una base de datos almacena el estatus de la estructura fisica de la base de datos. Estos estatus, guian a oracle durante la recuperación de disco.

El Redo Log File Online

Toda instancia de una base de datos tiene asociado un redo log file online para proteger la base de datos en caso de fallas en la instancia. Un redo log file online consiste en dos o más preasignaciones de archivos que almacenan los cambios hechos en la base de datos cuando estos ocurren.

Un redo log file online esta lleno con entradas redo. Las entradas redo almacenan datos que pueden ser usados para reconstruir todos los cambios hechos a la base de datos, incluyendo los segmentos de rollback almacenados en los buffers del SGA. Las entradas redo almacenan en un nivel bajo los cambios de la base de datos que no pudieron ser mapeadas por las acciones del usuario, asi mismo, el contenido de los redo log file online nunca deberian ser editados o alterados y nunca ser usados por alguna aplicación como las de auditoria.

Las entradas redo son almacenadas en forma circular en el redo log buffer del SGA y son escritas en el redo log file por medio del proceso background LGWR. Siempre que a una transacción se le hace un commit, LGWR escribe la entrada redo de la transacción en el redo log buffer del SGA y un "system change number" (SCN) es asignado para identificar la entrada redo para cada transacción.

Oracle requiere dos archivos para garantizar que uno esté siempre disponible para escritura mientras el otro esta archivado, si se desea. Cuando el actual redo log file es llenado, LGWR comienza a escribir en el siguiente redo log file online, cuando el ultimo redo log file disponible es llenado, LGWR retorna al primer redo log file online y escribe en este iniciando el ciclo nuevamente. La **figura 1** muestra la escritura circular de los redo log file online

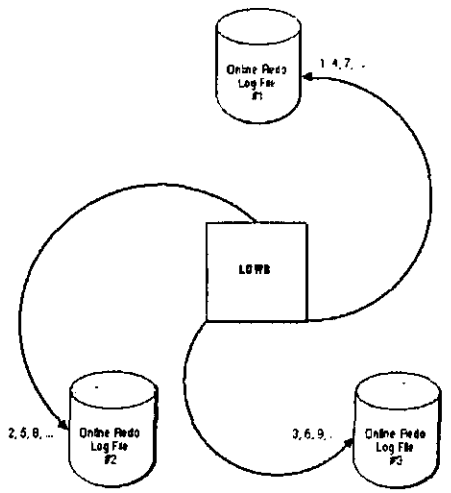


Fig. 1 Uso circular de los redo log files

Los redo log files online llenos están disponibles a LGWR para ser rehusados dependiendo si el archivo esta habilitado.

- Si el archivo esta deshabilitado, el redo log file lleno esta disponible una vez que el checkpoint este completado.
- Si el archivo está habilitado, los redo log file llenos están disponibles a LGWR una vez que el checkpoint esté completado y cuando haya sido archivado.

En cualquier momento, oracle usa solamente uno de los redo log files para almacenar entradas redo escribiendo desde el redo log buffer. El redo log file activo comienza a ser escrito por LGWR y es llamado el redo log file actual. Los redo log files online que se requieren para una recuperación de la instancia son llamados los redo log files online activos los demás son llamados inactivos.

Checkpoints

Un evento llamado checkpoint ocurre cuando un proceso background de oracle llamado DBWR escribe todas las modificaciones hechas en los database buffers en el SGA, incluyendo transacciones con commit y sin commit, en los datafiles. El checkpoint está implementado por las siguientes razones.

- Los checkpoints aseguran que los data segment en memoria que cambian frecuentemente sean escritos en los datafiles regularmente. Debido al algoritmo de el menos-recientemente-usado del DBWR, un data segment que cambia frecuentemente nunca calificaría como el bloque menos recientemente usado y entonces nunca sería escrito en el disco si el chekpoint nunca ocurre.
- Dado que todos los cambios en la base de datos han sido almacenados en los datafiles, las entradas redo antes del checkpoint no necesitan ser aplicadas en los datafiles si la recuperación de la instancia es necesaria, Debido a esto los checkpoint son usados por que estos pueden expedir recuperaciones de la instancia.

Oracle nunca para la actividad de las transacciones afectadas por el checkpoint, debido a esto, DBWR continuamente escribe de los database buffers al disco y el chekpoint no es necesariamente requiere muchos data blocks para escribir todo una sola vez.

Los checkpoint ocurren si se llenan o no los redo log files online y archivados. Si el archivo está deshabilitado un checkpoint afectando un redo log file online debe completarse antes de que el redo log file puede ser rehusado por LGWR. Los checkpoint pueden ocurrir para todos los datafiles de la base de datos (llamados database chekpoints) o pueden ocurrir solamente para datafiles muy específicos.

Los siguientes puntos explican cuándo un checkpoint ocurre y que sucede en cada situación.:

- Un checkpoint de base de datos ocurre automáticamente a cada momento que un log cambie. Si un checkpoint previo está actualmente en progreso, un nuevo checkpoint forzado se swichea sobre el checkpoint actual.
- Una inicialización de parámetro LOG_CHECKPOINT_ INTERVAL puede ser fijado para forzar al ckeckpoint de base de datos cuando un número predeterminado de redo log blocks ha sido escrito en el disco. Se puede fijar otro parámetro LOG_CHECKPOINT_TIMEOUT para forzar un checkpoint de base de datos a un número específico de segundos después de que le checkpoint ha sido iniciado. Estas opciones son usadas cuando los redo log files son extremadamente grandes y son usados checkpoint adicionales para switchear los logs.
- Cuando se inicia un respaldo de un tablespace online un checkpoint es forzado solamente sobre los datafiles que constituyen el tablespace que se ha comenzado a respaldar. Un checkpoint en el mismo momento se sobreescribe en el checkpoint previo en progreso. También desde este tipo de checkpoint solamente se afectan los datafiles que comenzaron a respaldarse y esto no reduce la cantidad de espacio que se necesitaría para una recuperación de la instancia.
- Si el DBA hace un teblespace offline con prioridad normal o temporal, un checkpoint es forzado solamente sobre los datafiles online del tablespace asociado.
- Si el DBA da de baja la instancia (NORMAL O INMEDIATA), oracle forza a un ckeckpoint para completar antes de que la instancia se de baja.

Mecanismos de un checkpoint

Cuando un checkpoint ocurre, el proceso background checkpoint recuerda la situación de la siguiente entrada para ser escrita en el redo log file online y le señala al DBWR que debe escribir las modificaciones de los database buffers de el SGA a los datafiles del disco duro. El CKPT entonces actualiza las cabeceras de todos los control files y datafiles para reflejar los últimos checkpoints.

Cuando un checkpoint no ocurre, el DBWR solamente escribe los últimos database buffers más recientemente usados al disco para liberar estos para nuevos datos. Sin embargo, como un checkpoint procede, el DBWR escribe datos en los datafiles sobre el nombre del checkpoint y sobre las operaciones de base de datos.

Dependiendo sobre que signos de el checkpoint ocurran, el checkpoint puede estar en modo normal o rápido. Con el modo normal del checkpoint, DBWR escribe un número pequeño de data buffers a cada momento. Con el checkpoint rápido, el DBWR escribe un gran número de database buffers cada momento y este ejecuta una escritura sobre el nombre del checkpoint.

Mientras un checkpoint es completado, todos los redo log files online escriben desde el último checkpoint y en caso de que la base de datos falle se necesita interrumpir el checkpoint y hacer una recuperación de la instancia necesariamente. Adicionalmente si el LGWR no puede accesar un redo log file online para escribir en el por que un checkpoint no ha sido completado, la operación de la base de datos se suspende temporalmente mientras el checkpoint completa su operación y el redo log file está disponible. En este caso el checkpoint es un checkpoint rápido asi, este se completa en cuanto le sea posible.

Redo log files online multiplexados

Oracle provee la capacidad de redo log files online multiplexados para respaldar contra posibles daños de los redo log files online. Con redo log files online multiplexados, el LGWR escribe concurrentemente la misma información log en múltiples e idénticos redo log files online, por eso se elimina un punto de falla de un redo log file. En la **figura 2** se observa un redo log file multiplexado y su forma de trabajar.

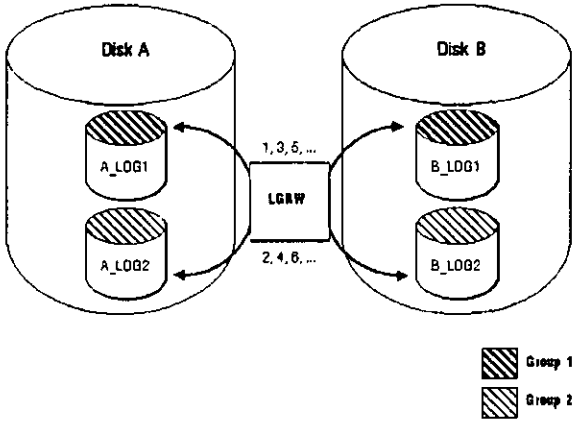


Fig. 2 Redo log file Multiplexado

Los correspondientes redo log file online son llamados grupos, cada redo log file en un grupo es llamada un miembro, cada miembro de un grupo esta concurrentemente activo (para ser escrito por LGWR). Si se usa la configuración de redo log files multiplexados, cada miembro en un grupo debe tener exactamente el mismo tamaño.

Mecanismo de trabajo de los redo log files multiplexados

LGWR siempre direcciona todos los miembros de un grupo, asi, el grupo contiene uno o más miembros, por ejemplo, LGWR siempre intenta escribir en todos los miembros de un grupo dado concurrentemente, entonces se swichea y concurrentemente escribe en todos los miembros de el siguiente grupo. LGWR nunca escribe concurrentemente en un miembro de un grupo dado y en un miembro de otro grupo.

LGWR reacciona de diferente manera cuando ciertos miembros del redo log file online no están disponibles, dependiendo de las siguientes razones:

- Si LGWR puede escribir con éxito en al menos un miembro en un grupo escribiendo para los miembros accesibles de el grupo procede en forma normal; LGWR simplemente escribe en los miembros disponibles de un grupo e ignora los miembros que no están disponibles.
- Si LGWR no puede accesar al siguiente grupo por que el grupo necesita ser archivado, la operación de la base de datos para temporalmente mientras el grupo está disponible.
- Si todos los miembros del siguiente grupo están inaccesibles para LGWR debido a que uno o más discos han fallado, un error es retornado y la instancia de la base de datos inmediatamente se da de baja. En este caso, la base de datos puede necesitar un media recovery debido a la pérdida de los redo log file online; sin embargo en el checkpoint ha sido movido más allá de los datos perdidos y el media recovery no sería necesario, simplemente elimina el grupo inaccesible.
- Si todos los miembros de un grupo de repente son inaccesibles para LGWR para que este comience a escribir, es retornado un error en la instancia e inmediatamente se da de baja. En este caso la base de datos podría requerir un media recovery para los redo log files online perdidos.

Siempre que LGWR no pueda escribir sobre un miembro de un grupo, oracle marca que el miembro debe ser reiniciado y mandar un mensaje de error en el TRACE file y el ALERT file indicará el problema con los archivos inaccesibles.

Para siempre estar protegidos contra la falla de un redo log file online, un redo log file multiplexado debería ser completamente simétrico: todos los redo log files online deberían tener el mismo número de miembros, sin embargo, oracle no requiere que un redo log file multiplexado sea simétrico; por ejemplo, un grupo puede tener solamente un miembro, mientras otros grupos pueden tener dos miembros. Oracle permite esta característica para prevenir situaciones que temporalmente afecten algún miembro de los redo log files online, pero pueden existir otras características como una falla de disco. En la **figura 3** podemos observar una configuración legal y una ilegal de redo log files online multiplexados.

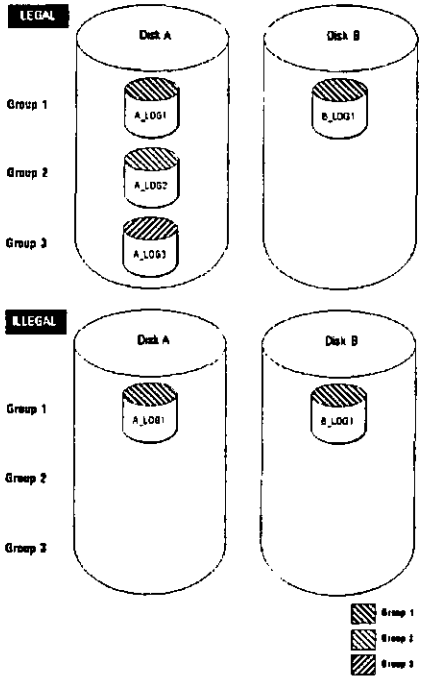


Fig. 3. Configuración legal e ilegal de redo logs multiplexados

Los Redo log files Archivados

Oracle permite la archivación opcional de grupos de redo log files online llenos de los cuales se crean redo log offline. La archivación de grupos llenos tiene dos ventajas con respecto a las opciones de respaldo y recuperación de la base de datos:

- Un respaldo de la base de datos junto con los redo log files online archivados, garantiza que todas las transacciones guardadas puedan ser recuperadas en caso de que ocurra un error de sistema operativo o falla de disco.
- En los respaldos online, mientras la base de datos este abierta y el sistema operativo se usa en forma normal puede ser usada si un los archivos son guardados permanentemente.

La elección de habilitar o no la archivación de grupos llenos de redo log files online depende del esquema de respaldo y recuperación para una base de datos oracle. Si no se puede producir para pérdida de cualquier dato en la base de datos en caso de que el disco falle, un log archivado debe estar presente. Sin embargo, la archivación de redo log files online puede requerir que el DBA ejecute operaciones de administración extra.

Mecanismos de Archivación

Dependiendo de cómo se configure la archivación, los mecanismos de archivación de grupos de redo log son ejecutados por el proceso background ARCH (en forma automática) o usar el proceso que usa un comando para archivar un grupo manualmente.

ARCH puede archivar un grupo una vez que el grupo esté inactivo y el swicheo de el siguiente grupo ha sido completado. El proceso ARCH puede accesar cualquier miembro de el grupo, necesario para completar el archivo del grupo. Por ejemplo, si ARCH intenta abrir un miembro de un grupo y este no está accesible, ARCH automáticamente prueba para usar otro miembro de el grupo y así mientras este encuentra un miembro de el grupo que esté disponible para archivar. Si ARCH está archivando un miembro de un grupo y la información de este grupo es inválida o hay una falla en el disco, ARCH automáticamente swichea otro miembro de el grupo para continuar.

Un grupo de redo log files online no puede estar disponible para LGWR para usarse mientras ARCH esté archivando el grupo. Esta restricción es importante por que garantiza que LGWR no escriba accidentalmente sobre un grupo que aun no ha sido archivado, lo cual previene el uso de todos los subsecuentes redo log files archivados durante una recuperación de la base de datos.

Cuando la archivación es usada, debe ser especificado un directorio de destino para almacenar separadamente al disco que contiene los datafiles, los redo log files y los control files de la base de datos. Generalmente el directorio destino es otro disco u otro servidor de base de datos.

Un redo log file archivado es una simple copia idéntica de los miembros llenos que constituyen un grupo redo log online, por consiguiente, un redo log file archivado incluye las entradas redo presentes en miembros idénticos de un grupo al momento que este grupo fue archivado. El redo log file archivado también preserva los números de secuencia de los grupos log.

Si la archivación está habilitada, LGWR no permite rehusar un grupo redo log mientras este se este archivando, debido a esto, se garantiza que los redo log archivados contengan una copia de cualquier grupo creado mientras la archivación estaba habilitada.

Control Files

El control file de la base de datos es un pequeño archivo binario necesario para iniciar la base de datos y operar exitosamente. Un control file es actualizado continuamente por oracle durante el uso de la base de datos, así que debe estar disponible para escritura siempre que la base de datos esté abierta. Si por alguna razón el control fiel no es accesible, la base de datos no podrá funcionar propiamente[1].

[1] Cada control file está asociado con una y solo una base de datos oracle.

Un control file contiene información acerca de la base de datos asociada que es requerida para accesar una instancia en una operación normal. La información de un control file puede ser modificada solamente por oracle; el administrador de la base de datos o los usuarios finales pueden editar el control file de la base de datos.

En todo momento que un datafile o un redo log file es agregado, renombrado o borrado desde la base de datos, el control file es actualizado para reflejar los cambios en la estructura fisica. Estos cambios son recordados para que:

- Oracle pueda identificar los datafiles y los redo log files online para abrirlos durante el inicio de la base de datos.
- Oracle pueda identificar archivos que son requeridos o que necesitan estar disponibles en caso de que sea necesaria una recuperación de la base de datos.

Debido a esto, si se hace algún cambio en la estructura fisica de la base de datos, inmediatamente se deberia hacer un respaldo de los control files.

Los control files también almacenan información sobre los checkpoints. Cuando un checkpoint inicia, el control file almacena información para recordar la siguiente entrada que deberia ser enterada en el redo log online. Esta información es usada durante la recuperación de la base de datos para decirle a oracle que todas las entradas almacenadas antes de este punto en el grupo redo log online no son necesarias para la recuperación de la base de datos.

Control files Multiplexados

Oracle permite múltiples e idénticos control files para ser abiertos concurrentemente y escritos por la misma base de datos. A través de almacenar múltiples control files por una sola base de datos sobre diferentes discos, se puede garantizar por si ocurre una falla con respecto a los control files. Si solo un disco contiene un control file y este falla, la instancia actual también falla cuando oracle intenta accesar al control file dañado, sin embargo, otras copias de el control file actual están disponibles en otros discos de tal suerte que una instancia pueda rastrear fácilmente sin necesidad de hacer una recuperación de la base de datos.

La pérdida permanente de todas las copias de los control files de la base de datos es un problema serio; si todos los control files de la base de datos se pierden permanentemente durante la operación (si todos los discos fallan), la instancia es abortada y se requerirá de un media recovery o recuperación de disco. El único camino para asegurar una rápida recuperación para una falla del sistema u otro desastre es un plan cuidadoso, se debe tener un conjunto de planes con procedimientos detallados.

RESPALDO DE LA BASE DE DATOS

Como parte de una estrategia para salvaguardar la base de datos de fallas potenciales que pueden dañar los datafiles y control files es necesario hacer respaldos de sistema.

Se pueden crear tres tipos de respaldos:

- Respaldos completos
- Respaldos parciales
- Utilerías de Export e Import

Respaldos completos

Un respaldo completo (full Backup) es un respaldo de sistema operativo de todos los datafiles y control files que constituyen la base de datos oracle. Un respaldo completo deberia también incluir los parameter files asociados a esta. Se puede hacer un respaldo completo ya sea con la base de datos abierta o cerrada, generalmente no se hace un respaldo completo después de que una instancia falla o bajo circunstancias inusuales.

Respaldos en Frío y Respaldos en Caliente

Después de dar de baja la base de datos, todos los archivos que constituyen la base de datos están cerrados y consistentes con respecto a el actual punto en el tiempo, así, se puede hacer un respaldo completo después de dar de baja la base de datos(respaldo en frío), así se puede usar para recuperar la información de ese punto en el tiempo en que se realizó el último respaldo. También se puede hacer un respaldo de la base de datos mientras esté abierta (respaldo en caliente) pero no consistente a un punto dado en el tiempo y se necesite recuperar.

Los datafiles obtenidos desde un respaldo completo son usados en algún tipo de esquema de media recovery[2]:

- Si la base de datos es operada en modo NOARCHIVELOG y el disco falla y se dañan algunos o todos los archivos que constituyen la base de datos, el más reciente respaldo completo puede ser usado para restaurar (no recuperar) la base de datos.

Debido a que un redo log archivado no esta disponible para levantar la base de datos en un punto del tiempo, todo el trabajo de la base de datos ejecutado desde el más reciente respaldo completo de la base de datos de tiene que repetir, bajo circunstancias especiales, un disco que falla en modo NOARCHIVELOG puede ser completamente recuperado pero es mejor no suponer este caso ideal.

- Si la base de datos opera en modo ARCHIVELOG y el disco falla y se dañan alguno o todos los archivos de la base de datos, los datafiles reunidos por el más reciente respaldo completo pueden ser usados como parte de una recuperación de la base de datos.

Después de restaurar los datafiles necesarios desde un respaldo completo, se puede continuar la recuperación aplicando los redo log files archivados y los que están en línea para restaurar los datafiles en un punto dado en el tiempo.

En resumen, si la base de datos opera en modo NOARCHIVELOG, un respaldo completo es solamente un método parcial para proteger la base de datos contra posibles fallas del disco; si la base de datos es operada en modo ARCHIVELOG, los archivos respaldados pueden ser usados para restaurar archivos dañados como parte de una recuperación de la base de datos cuando un disco falla.

Respaldo Parciales

Un respaldo parcial en un respaldo de sistema operativo pequeño con respecto a un respaldo completo hecho mientras la base de datos esté abierta o cerrada. Las siguientes líneas son ejemplos de respaldos parciales de la base de datos:

- Un respaldo de todos los datafiles para un tablespace individual.
- Un respaldo de un solo datafile
- Un respaldo de un control file

Los respaldos parciales son usados solamente para una base de datos que opera en modo ARCHIVELOG debido a que un redo log archivado está presente, los datafiles restaurados desde un respaldo parcial pueden ser consistentes con el resto de la base de datos durante el proceso de recuperación.

Respaldo de un Datafile

Un respaldo parcial incluye solamente algunos de los datafiles de la base de datos. De forma individual o un conjunto de datafiles específicos pueden ser respaldados independientemente de los otros datafiles, redo log files online y control files, se puede también respaldar un datafile ya sea que esté en línea o fuera de línea.

[2] Recordemos que un media recovery se realiza generalmente cuando la falla es generada por el disco duro.

La elección de hacer un respaldo en frío o en caliente[3] depende solamente de la disponibilidad de requerimientos de los datos, un respaldo en caliente se debe elegir solamente si los datos deben estar siempre disponibles.

Respaldo en frío de un datafile

Cualquier datafile de la base de datos puede ser respaldado cuando el archivo está fuera de línea o en línea, las siguientes situaciones son ejemplos de respaldos de datafiles:

- Una base de datos está dada de baja, como resultado, todos los datafiles de la base de datos están cerrados en forma normal o fuera de línea (offline). Si algún datafile de la base de datos dada de baja es respaldado entonces, este se considera un respaldo en frío de un datafile.
- Una base de datos está abierta y un tablespace está fuera de línea, como resultado, todos los datafiles de la base de datos están fuera de línea. Si cualquiera de los datafiles de un tablespace fuera de línea se respalda, este también es considerado como un respaldo en frío de un datafile.

Existen otras situaciones en las que se puede hacer un respaldo en frío de un datafile, sin embargo en ciertas circunstancias la base de datos puede estar dada de baja o el tablespace puede estar fuera de línea, pero los datafiles asociados pueden estar en línea con respecto al sistema operativo. Por ejemplo, un base de datos puede ser montada y cerrada para una recuperación de la base de datos, pero los datafiles asociados están abiertos y sufrir cambios durante la operación de recuperación. Se debe evitar entonces hacer respaldos de los datafiles en las situaciones dadas.

Cuando una instancia de la base de datos es dada de baja en prioridad normal (en otras palabras en modo no aborted) o cuando un tablespace está fuera de línea en modo normal , un respaldo en frío es una copia de datos consistentes. Todos los datos dentro del datafile fuera de línea respaldado es consistente con respecto a un punto en el tiempo (el momento en el que se hizo el respaldo).

Respaldos en caliente de los Datafiles

Si la base de datos está operando en modo ARCHIVELOG, se puede respaldar cualquier datafile mientras la base de datos esté abierta, mientras los tablespaces asociados estén en línea y mientras los datafiles están en línea actualmente en modo normal de uso. Este tipo de respaldo de datafile es considerado un respaldo en caliente de datafiles.

Un respaldo de un datafile en línea es una copia de datos inconsistente. Un datafile que esta en linera y comienza ser recuperado se dice estar incompleto o mal formado por que los bloques no necesariamente se escribieron en el orden que estos cambiaron, debido a esto, todos los datos dentro de un respaldo de un datafile en línea no garantiza consistencia con respecto a un punto en el tiempo, sin embargo un datafile incompleto o mal formado es fácilmente hecho consistente mediante procedimientos de recuperación de la base de datos.

Cuando se inicia el respaldo de un tablespace online (o de un datafile individual), oracle detiene el almacenamiento de ocurrencias de chekpoints en las cabeceras de los datafiles en línea respaldados. Esto significa que cuando un datafile es restaurado este tiene conocimiento de el más reciente checkpoint del datafile que ha ocurrido antes de que se respaldara el datafile en línea; como resultado, oracle pregunta por el conjunto de redo log files apropiado para hacer las tareas de recuperación de ser necesario. Una vez que un respaldo en caliente es completado, oracle avanza la cabecera del archivo de el checkpoint actual de la base de datos.

Datos Respaldados Consistentes e Inconsistentes

Pueden existir dos estatus en un datafile respaldado: datos consistentes e inconsistentes

Un respaldo de datos consistente es obtenido cuando el respaldo se hace sobre un datafile fuera de linera (respaldo en frío). Además, el datafile debe estar fuera de línea en forma normal y no como resultado de un error de I/O o haber sido puesto fuera de

[3] Cuando la base de datos está dada de baja o bien cuando esté funcionando normalmente, respectivamente

línea con la opción inmediate. Los datos en un simple archivo se dice están consistentes por que todos los bloques de datos corresponden a un determinado punto en el tiempo.

Para restaurar datafiles para un punto en el tiempo en particular, se puede usar un respaldo completo o parcial hecho mientras la base de datos estaba dada de baja los tablespaces estaban fuera de línea; debido a que los datos siempre están consistentes no se requiere tomar acciones para hacer que los datos en la restauración sean correctos.

Si la base de datos no se dio de baja limpiamente (por ejemplo, cuando una falla de la instancia ha ocurrido o bien se usó el comando SHUTDOWN ABORT) los datafiles fuera de línea pueden ser inconsistentes.

Se puede también usar un respaldo completo o parcial mientras la base de datos este abierta y los tablespaces en línea, para restaurar los datafiles en un punto en el tiempo en particular, sin embargo, los datos en el datafile restaurado son inconsistentes, debido a esto, los redo log files (archivados o en línea) deben ser reaplicados para hacer que los datos estén consistentes.

A continuación se muestra un ejemplo para entender como se puede generar un respaldo inconsistente y entonces usar una recuperación de la base de datos.

Ejemplo:

Se hace un respaldo de un datafile que esta en un tablespace online. El datafile correspondiente tiene 4 datablocks. Para simplificar y hacer más sencillo el entendimiento, un bloque de datos es representado por una letra **Fig. 4.**

1- Al primer momento, el bloque 1 de el datafile es escrito en el archivo de respaldo
2- Al segundo instante, el bloque 2 de el datafile es escrito en el archivo respaldado, al mismo tiempo una versión modificada del bloque 1 es escrita desde el SGA al datafile.
3- En un tercer instante, el bloque 3 de el datafile es escrito en el archivo de respaldo
4- En el cuarto instante una versión modificada del bloque 4 es escrita desde el SGA al datafile y este bloque modificado es escrito en el archivo de respaldo

Al menos dos entradas redo son generadas debido a las modificaciones del los bloques 1 y 4

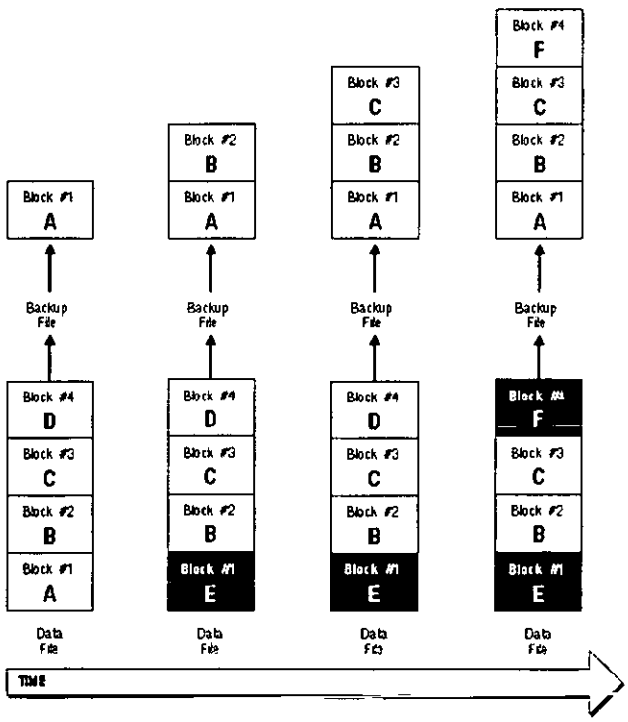


Fig. 4. Ejemplo de un archivo respaldado en caliente

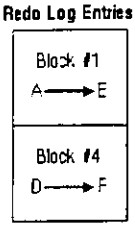


Fig. 5. Entradas redo generadas durante el respaldo en caliente

Si el archivo de respaldo es restaurado, el datafile restaurado estará inconsistente, sin embargo, cuando la recuperación se ejecute desde el correspondiente redo log file los datos entonces estarán consistentes.

Respaldo de Control Files

Otra forma de un respaldo parcial es respaldar un control file, debido a que el control file tiene referencias de la base de datos asociada con respecto a su estructura física, un respaldo de un control file debe ser hecho continuamente, los control files multiplexados pueden salvaguardar contra pérdida de un solo control file, sin embargo, si un disco falla y daña los archivos, entonces esté método no substituye al respaldo de los control files.

Las Utilerías de Export e Import se usan para mover datos de oracle fuera de la base de datos. Export es una utilería que escribe datos desde una base de datos oracle a archivos del sistema operativo en formato oracle. Los archivos export almacenan información del esquema de los objetos creados por la base de datos. Import es una utilería que lee los datos de archivos export y restaura la información correspondiente dentro de la base de datos existente.

Las utilerías de export e import se usan para:

- Salvar definiciones de tablas o datos en un mismo punto en el tiempo (Archivo histórico)
- Salvar definiciones de tablas (Con o sin datos) fuera de línea con la intención de regresar esta en una base de datos tiempo después
- Mover datos desde una base de datos sobre una máquina a otra base de datos sobre otra máquina
- Mover Datos entre dos versiones de software de oracle server.[4]
- Proveer protección de aplicaciones en caso de una falla, permitiendo restaurar una tabla o un conjunto de tablas al tiempo que el export fue ejecutado sin requerir que la base de datos completa sea recuperada.
- Reorganizar tablas para eliminar registros migrados y fragmentación.

RECUPERACION DE LA BASE DE DATOS

En toda base de datos la posibilidad de que el sistema falle siempre está presente. Si el sistema falla se debe recuperar la base de datos rápidamente y de esta manera el impacto en los usuarios sea mínimo.

Para hacer una recuperación de algún tipo de falla de sistema requiere lo siguiente:

1- Determinar cuales estructuras de datos están intactas y cuales necesitan recuperación
2- Seguir los pasos apropiados de recuperación.
3- Reiniciar la base de datos de tal suerte que pueda reasumir su operación normal.
4- Asegurar que no se ha perdido ningún dato y que no entren datos incorrectos en la base de datos.

[4] Para migrar o actualizar una base de datos se puede utilizar la utilería de export de la versión actual de la base de datos y cargar la nueva base de datos con la utilería import. Para asegurar la consistencia del export, hay asegurar que la base de datos no esté disponible para actualizaciones después de usar export, así, la nueva versión de la base de datos estará completa

La meta es retornar lo más rápidamente posible la operación normal. Los procesos de recuperación varian dependiendo del tipo de falla y de archivos de la base de datos que fueron afectados por las fallas.

Caracteristicas de la Recuperación

Oracle ofrece diferentes caracteristicas para proveer facilidad en las estrategias de recuperación:

- Recuperación de sistema, software o fallas de hardware
- Recuperación automática de instancia al levantarse la base de datos
- Recuperación de tablespaces individuales o de archivos mientras el resto de la base de datos esta operando
- Operaciones basadas en tiempo o en cambios para recuperar una transacción consistente.
- Incrementar el control sobre el tiempo de recuperación en el caso de que el sistema falle.
- La habilidad de aplicar entradas redo log en paralelo para reducir la cantidad de tiempo de la recuperación.
- Utilerías de Import y Export para archivar y restaurar datos en un formato lógico en lugar de un respaldo físico de archivos.

Database buffers y DBWR

Los database buffers en el SGA son escritos al disco solamente cuando es necesario, usando el algoritmo del menos usado recientemente. Debido al camino que DBWR usa para escribir los database buffers a los datafiles, estos podrian contener algunos data blocks modificados y sin guardar aun sus transacciones

Pueden resultar dos problemas potenciales cuando ocurre una falla en la instancia:

- Los data blocks modificados por una transacción pueden no ser escritos en los datafiles en el momento de hacer el commit y pueden solamente aparecer en los redo log, debido a esto, los redo log contienen cambios que podrian ser reaplicados en la base de datos durante la recuperación.
- El redo log también puede contener datos que no se guardaron, los cambios de las transacciones no guardadas aplicadas por el redo log deben ser borrados de la base de datos.

Para resolver estos problemas existen dos pasos separados que generalmente se usan con éxito para recuperar fallas del sistema: rollin forward con el redo log y rolling back con los segmentos de rollback.

El redo log y rolling forward

El redo log es un conjunto de archivos del sistema operativo que almacena todos los cambios hechos para algún database buffer, incluyendo datos, indices y segmentos de rollback, donde los datos pueden haber sido guardados o no guardados. El redo log protege cambios hechos a los database buffers en memoria que no han sido escritos en los datafiles.

El primer paso de recuperación de una instancia o falla de disco es hecho por el roll forward, que consiste en reemplazar todos los cambios almacenados en el redo log a los datafiles. Debido a esto, los datos recuperados también son almacenados en el redo log, rollin forward también regenera los segmentos de rollback correspondientes.

El primer paso de recuperación es el roll forward, que es una replicación a todos los datafiles de los cambios que se almacenados en los redo log files . Si toda la información necesaria está en linea, incluyendo datos, indices, y segmentos de rollback, oracle ejecuta esta recuperación automáticamente cuando la base de datos se levanta. Después de esto, los datafiles contendrán los datos a los que se les aplicó commit asi como a los que no se les aplicó y que estaban almacenados en los redo log files. El redo log protege por cambios hechos a database buffers en memoria que no han sido escritos en los datafiles.

Los primeros pasos para la recuperación de una instancia en caso de error del disco es hacer un roll forward, o volver a aplicar todos los cambios grabados en los redo log a los datafiles y luego deshacer en un segundo paso aquellas transacciones que no hayan sido confirmadas

Segmentos de Rollback y Rolling Back

Los segmentos de rollback almacenan acciones de la base de datos que se deberían deshacer durante ciertas operaciones de la base de datos. En una recuperación de la base de datos los segmentos de rollback deshacen los efectos de transacciones no guardadas previamente, aplicadas por la fase de rolling forward.

Después del roll forward, algunos cambios que no estaban guardados deben ser deshechos. Después los redo log files tienen que reaplicar todos los cambios hechos en la base de datos, entonces los correspondientes segmentos de rollback son usados. Los segmentos de rollback son usados para identificar y deshacer las transacciones que nunca fueron salvadas y que ya estaban almacenadas en los redo log y aplicadas para la base de datos durante el roll forward. Este proceso es llamado rolling back, La **figura 6** muestra el rolling forward y el rolling back, los dos pasos necesarios para recuperar información cuando ocurre algún tipo de falla del sistema.

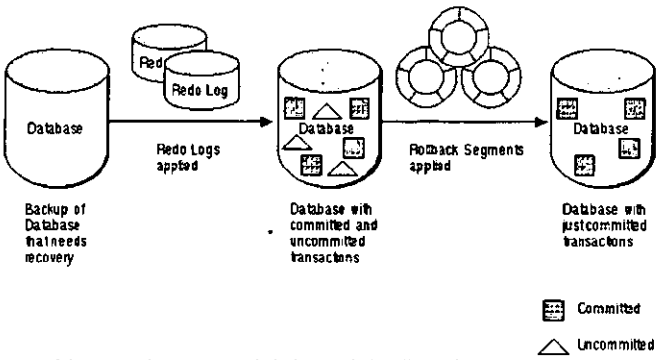


Fig. 6. Muestra los pasos básicos del rolling forward y rolling Back

Rolling Forward y Rolling Back

Oracle puede hacer rollback a múltiples transacciones simultáneamente como sea necesario, todas las transacciones que estén activas al momento de la falla son marcadas como "muertas". En lugar de esperar, SMON hace roll back sobre las transacciones muestras y las nuevas transacciones pueden recuperar bloqueando transacciones. Esta característica es llamada fast transaction rollback

Recuperación para fallas de instancia

Cuando una instancia es abortada en forma inesperada (por ejemplo un proceso background que falle), o en forma esperada (por ejemplo cuando se usan los comandos SHUTDWN ABORT o STARTUP FORCE), una falla en la instancia ocurre y una recuperación de la instancia es requerida. La recuperación de la instancia restaura la base de datos a su estado de transacción consistente justo antes del fracaso anterior.

Si se experimenta una falla durante la realización de un respaldo en caliente, entonces se requeriría de un media recovery. En otro caso, oracle automáticamente ejecuta la recuperación de la instancia para la base de datos una vez que la base de datos es reiniciada (montada y abierta por una nueva instancia).

Para la transición de un estado de montado a un estado de abierto se siguen esos pasos:

1- Se hace el rolling forward para recuperar datos que no han sido almacenados en los datafiles y que ya has sido almacenados en los redo log online incluyendo el contenido de los segmentos de rollback.
2- Abrir la base de datos. En lugar de esperar para que todas las transacciones sean deshechas antes de hacer la base de datos disponible, oracle habilita la base de datos para ser abierta tan pronto como el cache recovery es completado. Algunos datos que no son cerrados por transacciones que no se podrían recuperar están inmediatamente disponibles. Esta característica se llama "fast warmstart".
3- Hacer todas las transacciones que estén activas al momento de la falla como "muertas" y marcar los segmentos de rollback que contienen estas transacciones como parcialmente disponibles.
4- Recuperar transacciones muertas como parte de la recuperación del SMON
5- Resolver cualquier transacción pendiente pendiente al momento de sufrir la falla de la instancia.

Tablespaces de solo lectura y Recuperación del Instancia

No es requerida una recuperación de los data files de solo lectura después de una recuperación de la instancia. La recuperación durante el inicio verifica que un archivo de solo lectura no necesita media recovery. Esto es por que el archivo no fue restaurado por desde un respaldo y este era de solo lectura. Si se restaura un tablespace de solo lectura desde un respaldo, no se tiene acceso al tablespace mientras no se complete el media recovery.

Recuperación de Falla de disco (Media Failure)

El Media failure es una falla que ocurre cuando un archivo o una porción del archivo o el disco no pueden ser leídos o escritos debido a que están dañados o perdidos. Por ejemplo, esto puede pasar si uno o más datafiles son borrados accidentalmente o perdidos por fallas o errores en el disco.

La recuperación para un media failure puede hacerse de dos formas distintas, dependiendo sobre el modo de Archivar en el cual la base de datos esté operando (ARCHIE LOG o NOARCHIVELOG):

Si la base de datos esta operando de tal forma que los redo log online son solamente rehusados y no archivados, la recuperación para un media failure es una simple restauración desde el respaldo más reciente completo. Todo el trabajo hecho después de hacer el respaldo debe hacerse manualmente de nueva cuenta

Si la base de datos esta operando de tal forma que los redo log files están archivados, la recuperación para un media failure puede ser hecha por un procedimiento para reconstruir la base de datos dañada y así tener transacciones consistentes igual al estado antes del media failure

La recuperación para un media failure siempre recupera la base de datos entera y procura tener transacciones consistentes. Si no es posible recuperar parte de la base de datos (tal como un tablespace) en un punto en el tiempo, y se pudiera recuperar otra parte de la base de datos de un punto diferente en el tiempo, entonces la base de datos no tendría transacciones consistentes.

A continuación se explican los diferentes tipo de media recovery disponibles si la base de datos está operando en modo ARCHIVELOG:

Tablespaces de solo lectura y Media Recovery

Un media recovery normal no checa los estatus de solo lectura de los datafiles. Cuando se ejecuta un media recovery de un tablespace de solo lectura se tienen tres posibles opciones, dependiendo de cuando el tablespace fue puesto a solo lectura y cuando se ejecutó el más reciente backup, podemos ver esto en la **figura 7**.

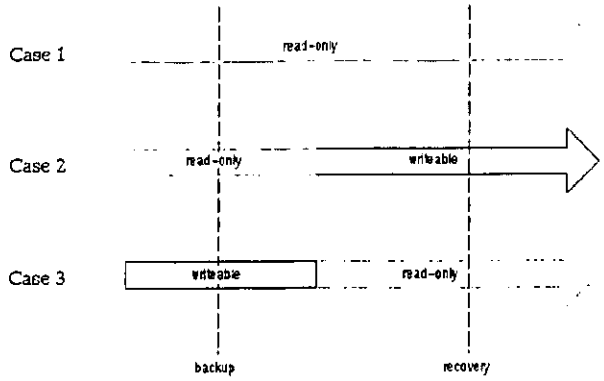


Fig. 7. Tipos de Media Recovery.

Caso 1

El tablespace comienza a recuperarse si es solo lectura y fue solo lectura cuando se ejecutó el último respaldo. En este caso se puede simplemente restaurar el tablespace desde el respaldo y no será necesario aplicar información complementaria.

Caso 2

El tablespace que comienza a recuperarse es de escritura pero fue de solo lectura cuando se hizo el último respaldo; en esta circunstancia, se necesitaría restaurar el tablespace desde el respaldo y aplicar información complementaria (recovery) desde el punto en el tiempo en que el tablespace se hizo de lectura.

Caso 3

El tablespace que comienza a recuperarse es solo lectura pero era de escritura también al momento de hacer el último backup. Debido a que se debería respaldar un tablespace después de hacerlo de solo lectura, cuando esto ocurre se debe restaurar el tablespace desde el respaldo y hacer una recuperación al tiempo en que el tablespace fue hecho solo lectura.

Tanto los datafiles de escritura y solo lectura no pueden quedar fuera de línea automáticamente si un ocurre un media failure; si se experimenta un media failure que afecte solamente una porción de los datafiles se deberían tomar los datos de los datafiles offline para ejecutar una recuperación de los tablespaces offline en una base de datos abierta.

Media Recovery Completo

Un media recovery completo es una recuperación de todos los cambios perdidos. El media recovery completo es posible solamente si todos los redo logs (en línea y archivados) necesarios están disponibles. Existen diferentes tipos de media recovery completos, dependiendo de los archivos que están dañados y la disponibilidad de la base de datos que es requerida durante la operaciones de recuperación.

Recuperación con la base de datos Cerrada

Es un media recovery completo de todos los datafiles individuales que han sido dañados y puede proceder mientras la base de datos esta montada pero cerrada y completamente no disponible para su uso normal. Este tipo de recuperación es usada para los siguientes casos:

- La base de datos no puede ser abierta (la porción no dañada de la base de datos no está disponible para su uso).
- Los archivos dañados en el disco que falló incluyen uno o más datafiles que constituyen el tablespace de system o el tablespace que contiene segmentos de rollback activos.

Recuperación de un tablespace fuera de línea con la base de datos abierta

Un media recovery completo puede proceder mientras la base de datos esté abierta, los tablespaces que no han sido dañados en la base de datos están en línea y disponibles para su uso, mientras los tablespaces dañados están fuera de línea y todos los datafiles que constituyen los tablespaces dañados serán recuperados como una unidad. Los tablespaces offline recuperados son usados en la siguiente situación:

- Tablespaces no dañados de la base de datos que deben estar disponibles para su uso normal
- Archivos dañados por la falla del disco que no incluyen algún datafile que sea parte del SYSTEM tablespace o tablespaces que contengan segmentos de rollback activos

Recuperación de un datafile individual con la base de datos abierta y tablespace online

Para este caso también puede proceder con la base de datos abierta, los tablespaces que no fueron dañados de la base de datos están en línea y disponibles para su uso, mientras que los tablespaces dañados están fuera de línea y los datafiles dañados que están asociados con los tablespaces dañados serán recuperados bajo las siguientes situaciones:

- Los tablespaces que no fueron dañados deberian estar disponibles para su uso normal
- Los archivos dañados por la falla del disco no incluyen algún datafile que sea parte del SYSTEM tablespace o tablespaces que contengan segmentos de rollback activos

Media recovery completo usando un respaldo de el control file

Un media recovery completo puede proceder sin datos perdidos, aun cuando todas las copias de los control files estén dañadas por la falla del disco, un media recovery de los respaldos de los datafiles puede hacerse si el control file esta respaldado. El control file no es recuperable por un media recovery, más bien el RESETLOGS de una base de datos abierta pueden recuperar el control file.

Ejemplo del mecanismo de un media recovery completo

El mecanismo que oracle usa para ejecutar algún tipo de media recovery completo es descrito de mejor forma usando un ejemplo. Suponemos que se hará un media recovery completo de datafiles dañados mientras la base de datos esta abierta y un tablespace esta fuera de línea. Asumiremos los siguientes puntos:

- La base de datos tiene tres Datafiles:
- USERS1 Y USERS2 son los datafiles que constituyen el tablepace USERS almacenado en el disco X del servidor de base de datos.
- SYSTEM es el datafile que constituye el tablespace de SYSTEM, almacenado en el disco Y de el servidor de base de datos
- El disco X de la base de datos se cae
- Los redo lo file online comienzan a escribir al momento que el disco falla tienen el número de secuencia 31
- La base de datos trabaja en modo ARCHIVELOG.

La recuperación de dos datafiles que constituyen el tablespace de USERS es necesaria por que el disco X se ha dañado y automáticamente el tablespace se ha puesto fuera de línea. En este caso el tablespace de SYSTEM no ha sido dañado, debido a esto, la base de datos puede abrirse con el tablespace de SYSTEM online y disponible para su uso mientras la recuperación es completada.

Fase 1.- Restauración de los Datafiles Respaldados

Después que el disco 1 ha sido reparado, el más reciente respaldo de archivos es usado para restaurar solamente los datafiles dañados USERS1 y USERS2. Después de la restauración, los datafiles de la base de datos vuelven a existir, tal como se muestra en al **figura 8**.

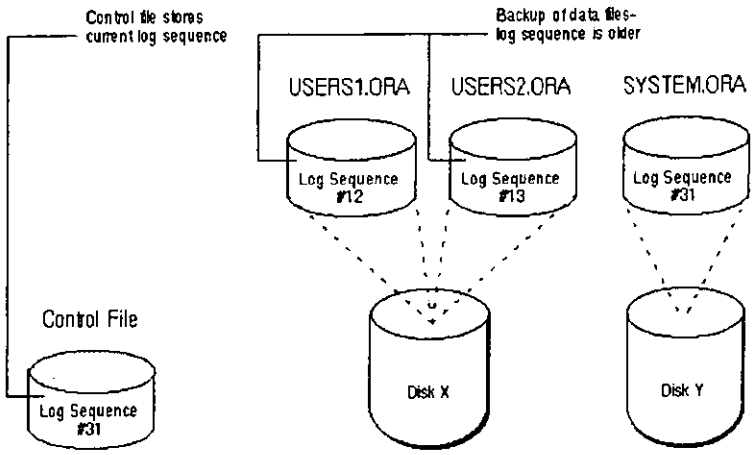


Fig. 8. Fase 1 del media recovery completo

Cada cabecera de datafile contiene el más reciente número de secuencia log. El control file contiene un apuntador para el último número de secuencia log que ha sido escrito.

Fase 2.- Rolling Forward con el Redo log

Cuando se completa el media recovery, oracle aplica los redo log files (archivados o en linea) a los datafiles como sea necesario, como se muestra en la **figura 9**, oracle automáticamente detecta cuando un redo log file no contiene información para restaurar que corresponda a un datafile restaurado, debido a esto, oracle optimiza el proceso de recuperación pero no intenta aplicar el redo log file para el datafile restaurado.

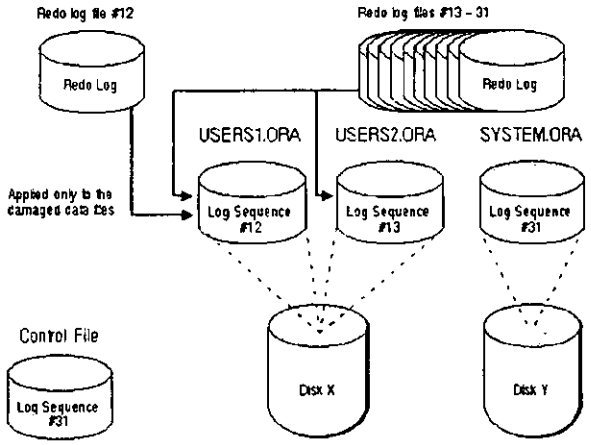


Fig. 9 Fase 2 de media recovery completo

En este caso, el redo log file con el número de secuencia log 12 es aplicado exclusivamente para USERS1, y los redo log files con los números de secuencia entre 13 y 31 son aplicados para ambos USERS1 y USERS2. No se aplican los redo log files para los datafiles que no requieren recuperación.

Esta es una bandera en la cabecera de el actual redo log que indica si este último redo log file esta disponible para aplicarse a los datafiles restaurados.

Fase 3.- Rolling back usando segmentos de rollback

Una vez que los redo log files necesarios han sido aplicados a los datafiles dañados, todos los datos no guardados que existan como resultado del roll forward en la fase 2 deben ser removidos.

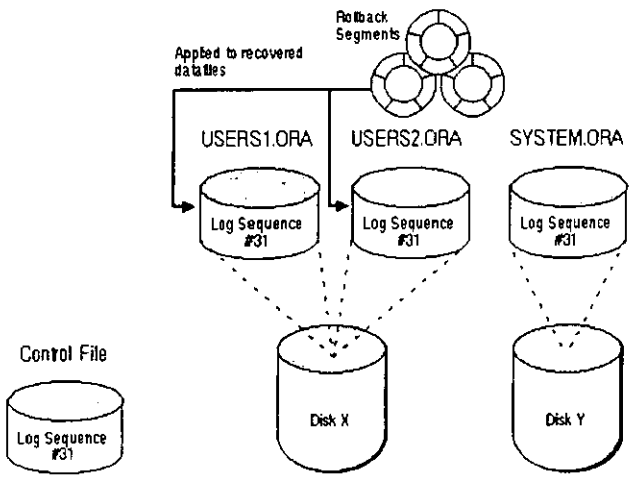


Fig. 10. Fase 3 del media recovery completo

Después que la fase 3 es completada, hay que notar como el número de secuencia log esta contenido en las cabeceras de los datafiles previamente dañados y restaurados USERS1 y USERS2, han sido actualizados durante la fase 2 del proceso de recuperación. El tablespace de USERS puede ahora ser puesto en línea. Los segmentos de rollback diferidos son aplicados a los datafiles del tablespace fuera de línea cuando este vuelva a estar en línea. Una vez que el rollback este completo, los datafiles USERS1 y USERS2 existen como eran instantes antes de que el disco fallara. Cuando este proceso termina, todos los datos en el tablespace son ahora consistentes y disponibles para su uso.

Media recovery incompleto

En situaciones especificas (por ejemplo, la perdida de todos los redo log files activos o un error de usuario tal como el borrado accidental de una tabla importante), un media recovery completo y puede no ser posible o puede no ser deseado. En esta situación, un media recovery incompleto es ejecutado para reconstruir la base de datos dañada para un estado transacciones consistentes como antes de que ocurriera el error.

En muchos casos, a menos que se deseé, un media recovery incompleto no es necesario si el redo log en linea ha sido multiplexado para proteger por futuras fallas.

Existen diferentes tipos de media recovery incompleto que pueden ser usados, dependiendo de la situación que se requiera: basado en cancelación, basado en el tiempo y basado en cambios.

Recuperación basada en Cancelación

En algunas situaciones, un media recovery incompleto debe ser controlado tal que el DBA pueda cancelar la operación en un punto especifico. Específicamente, la recuperación basada en cancelación es usada cuando uno o más grupos redo log (en línea o archivados) han sido dañados por la falla del disco y no están disponibles para los procedimientos requeridos de la recuperación (por ejemplo, los redo log en línea no tienen espejo y un solo redo log file en línea activo ha sido dañado por la falla del disco). Si uno o más grupos de redo log no están disponibles, los grupos de redo log perdidos no pueden ser aplicados durante los procedimientos de recuperación, debido a esto, el media recovery debe ser controlado tal que la operación de recuperación sea terminada después de que el más reciente y no dañado grupo redo log ha sido aplicado a los datafiles.

Recuperación basada en tiempo y basada en cambios

Un media recovery es deseable si el DBA necesita recuperar información de un punto especifico en el pasado. Este podría ser usado en las siguientes situaciones:

- Cuando un usuario accidentalmente borra una tabla y se da cuenta del tiempo aproximado cuando este error fue registrado. El DBA puede inmediatamente dar de baja la base de datos y recuperar esta en un punto en el tiempo justo antes de ocurrir el error de usuario.
- Cuando parte de un redo log file en línea se corrompe provocando una falla de sistema, de esta manera, el redo log file en línea activo está de pronto no disponible, la base de datos es abortada y un media recovery es requerido al instante. Las entradas redo en el redo log file en línea usado más recientemente son validas hasta el lugar donde se escribieron los datos corruptos y los datos que entraron después son inválidos. Solamente la parte dañada de el actual redo log file puede ser aplicada. En este caso, el DBA puede usar la recuperación basada en tiempo para detener los procedimientos de recuperación una vez que la porción válida de el más reciente redo log file en línea ha sido aplicada a los datafiles.

En ambos casos, el punto final de un media recovery incompleto puede ser especificado por un punto en el tiempo o por el número especifico de cambios en el SISTEMA (SCN). Un SCN es un registro en el redo log, en las entradas redo, cada que una transacción es guardada. Si es dado un tiempo, la base de datos es recuperada justo antes del SNC especificado.

Los mecanismos de una recuperación incompleta de la base de datos

Los procedimientos de recuperación de la base de datos incompleta es igual que para el media recovery completo con algunas excepciones:

- Todos los datafiles deben ser restaurados usando archivos respaldados completados antes de la recuperación (los archivos podrían venir incluso de diferentes respaldos ya sea parciales o hechos en tiempos diferentes).
- Para mejores resultados, el control file usado durante la media recovery incompleto debería reflejar la estructura física de la base de datos al tiempo de la recuperación. Si se abre la base de datos usando la opción de RESETLOGS, o si se abre la base de datos usando un comando CREATE CONTROLFILE, oracle checa el control file con el diccionario de datos actual. Si algún data file ha sido agregado o borrado del diccionario de datos, oracle actualiza el control file. Algunas otras diferencias son reportadas con mensajes de error.
- La recuperación puede terminar antes de que todos los redo log files disponibles sean aplicados. Las operaciones de recuperación pueden ser canceladas manualmente o estas pueden terminar automáticamente cuando el punto de parada llegue.
- Si el log de la base de datos es reiniciado como parte de un media recovery incompleto, todos los tablespaces que contienen los datafiles fuera de línea (durante la recuperación incompleta) deben ser borrados a menos que el tablespace fuera puesto en línea normalmente.
- Si un media recovery incompleto es actualmente una recuperación completa (por ejemplo, todos los redo log disponibles fueron aplicados por que un tiempo futuro en un SCN fue especificado), la base de datos debe ser abierta sin resetear la secuencia log. Sin embargo después que un media recovery incompleto es terminado, el número actual de secuencia log de la base de datos puede ser reiniciado a 1. Esta operación invalida las entradas redo presentes en todos los redo log files en línea y redo log files archivados. Después que la secuencia es reiniciada, los log de la base de datos (ambos, en línea y

archivados) existen como si estos fueran creados en forma reciente y los redo log files en linea no contienen entradas redo aun. Esta operación se muestra en la **figura 11**.

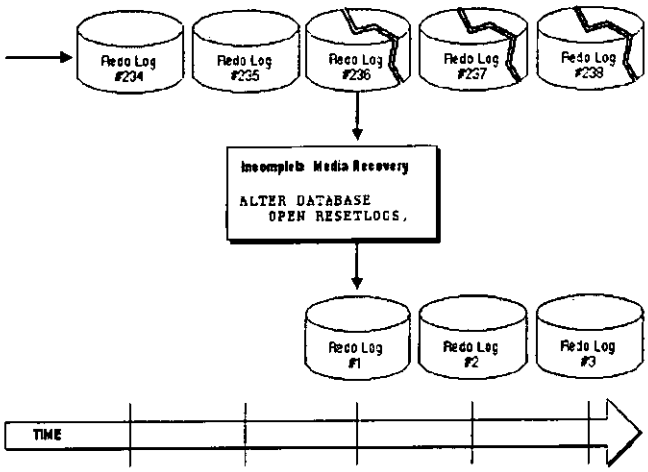
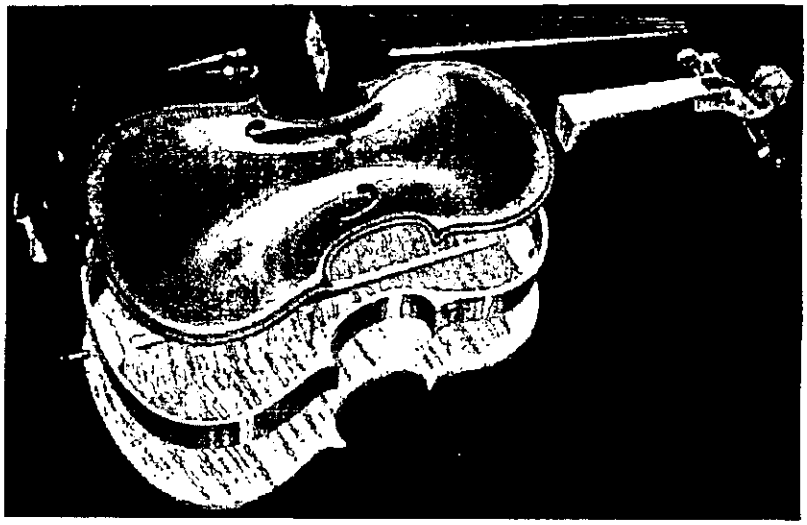


Fig. 11. Efectos de reiniciar el número de secuencia log en un media recovery incompleto.

El último paso pero no menos importante que los demás es poner a punto la base de datos, para ello podemos ayudarnos de las técnicas y utilerías para la afinación de la base de datos oracle, entre algunas utilerías podemos tener el tkproff, el analize y así también el explain plan, estas y otras que se complementan entre sí son analizadas en el capitulo siguiente.

Capítulo IV

Afinación de la Base de Datos



Niveles de Crisis

El principal problema que tiene casi cualquier base de datos es su constante cambio. Los cambios pueden darse de distintas formas, por ejemplo, las tablas de la base de datos crecen en cantidad de registros, las instancias crecen en número y las aplicaciones pueden estar desde un inicio mal distribuidas. Cada cambio puede ser la causa para entrar en un modo de crisis sobre el rendimiento.

El nivel de la crisis puede ser cuantificable si se mide cuánto tiempo gasta el administrador del sistema. Si supervisa constantemente, modifica y afina el sistema durante noches y fines de semana, el nivel de la crisis es alto. Si existe poco o ningún esfuerzo en la administración los sistemas, el nivel de la crisis es muy bajo.

El nivel de crisis es alto cuando se da inicio en el uso de alguna aplicación. Cuando los usuarios empiezan a familiarizarse con la aplicación el nivel de crisis disminuye y como los usuarios empiezan a tener un dominio mayor en el uso del sistema el nivel de errores que se puedan generar tiende a disminuir.

Los usuarios insertan datos en el sistema y por consiguiente el tamaño de la base de datos aumenta en forma exponencial. Esto genera un impacto directamente al tiempo de respuesta, siendo este un parámetro necesario para cumplir con la ejecución de los querys más comúnmente usados en el sistema. Mientras más y más registros son agregados al sistema por los usuarios, el tiempo de respuesta aumenta hasta llegar a ser casi inaceptable. Los querys comienzan a tener tiempos muy largos en su ejecución y el hardware probablemente tampoco soporte esos procesos tan pesados pero a la vez necesarios para la aplicación.

Actualización de hardware y afinación

La necesidad de actualizar el hardware es para mejorar el rendimiento del sistema. Una actualización de hardware reduciría el nivel de crisis en lo que respecta al soporte de hardware para la aplicación. Una alternativa podría ser por ejemplo incrementar la cantidad de microprocesadores en el servidor que se usa actualmente, usando sistemas avanzados de I/O y agregar memoria al servidor, cada una de las alternativas tiene un costo y un beneficio en la mejora del funcionamiento de la aplicación.

Sin embargo, la actualización de hardware podría no representar un buen impacto con respecto al tamaño de la base de datos. Para esta situación podría elegirse incrementar el tamaño asignado a la base de datos (por ejemplo asignar espacio a las áreas de los tablespaces temporales), pero sin alterar el rango en el cual los registros son insertados en la DB.

Adicionalmente a la actualización de hardware, también podría afinarse el entorno de oracle ya sea incrementando el espacio disponible en memoria y ordenando las áreas dentro de la base de datos.

Como el tamaño de la base de datos se incrementa, el tiempo de respuesta a corto plazo tendría que mejorar. El nivel de crisis debe decrementarse o bien estabilizarse. Cuando una nueva parte del sistema es puesta en producción el nivel de crisis se incrementa. A menudo no es posible que el total de volumen de información del sistema no pueda ser usado debido a que no se tiene un adecuado modelo de producción y el funcionamiento del sistema deberá ser revisado. Una forma de mejorar esta situación es la afinación de los querys de tal forma que queden en un nivel de respuesta aceptable. También para lograr todo esto debemos poner especial cuidado en tres áreas críticas:

1. Un Plan para el crecimiento de la base de datos. Se deben crear las aplicaciones considerando el crecimiento de la base de datos tomando en cuenta a la comunidad de usuarios y los desarrolladores de las aplicaciones
2. Afinación efectiva. Se debe ser capaz de identificar y resolver problemas de afinación causados durante el uso y crecimiento del sistema
3. Actualizaciones. Se debe estar familiarizado con las últimas versiones de oracle y la tecnología del hardware de tal forma que siempre se obtenga la mejor ventaja durante la afinación del sistema.

Incremento al nivel de crisis

El nivel de crisis se puede incrementar por diferentes causas incluyendo las de requerimientos extraordinarios. La razón principal por la que podría incrementarse el nivel de crisis es por cambios que aumentan dramáticamente el tiempo de respuesta. Por ejemplo, podríamos considerar un sistema que inicie su operación como una pequeña aplicación y más tarde la empresa sea absorbida por una empresa mucho más grande. Como resultado del cambio los requerimientos de la aplicación crecen para poder incluir las nuevas características de la nueva empresa.

Generalmente los cambios en los requerimientos de un sistema son implementados para agregar nueva funcionalidad a los sistemas existentes sin hacer un análisis de impacto sobre el diseño original, de tal suerte que no se prevé el impacto en el tiempo de respuesta. Como el sistema no esta completamente remodelado, tampoco sabemos el impacto sobre el rendimiento y esto puede resultar en problemas en los tiempos de respuesta.

Otro problema fuerte pero finalmente humano es la disponibilidad del personal de desarrollo, se necesitan personas que entiendan los requerimientos del negocio que el usuario necesita, que tengan experiencia y sobre todo que estén al día con las nuevas tecnologías. Por ejemplo, para soportar el constante crecimiento de la base de datos, necesitaríamos un DBA que tenga experiencia en el manejo de da bases de datos que varían constantemente así como tener el equipo apropiado para resolver problemas de afinación.

Después de afinar realmente un sistema oracle, se pueden observar un modelo en el cual los problemas de rendimiento disminuyan. Algunos de los factores más comunes que pueden reducir el nivel de crisis son los siguientes:

Recursos

Los recursos tales como CPUs, memoria, capacidad de entrada/salida y el ancho de banda de la red son las llaves principales para reducir el tiempo de respuesta. Agregando este tipo de recursos al sistema se hace posible que sea más veloz el tiempo de respuesta y el rendimiento depende de los siguiente:

- Qué tantos recursos están disponibles
- Qué tantos clientes necesitan los recursos
- Qué tan grande debería ser la espera por estos recursos
- En qué medida se pueden mantener estos recursos

La **figura 1** muestra como se incrementa el nivel de uso dependiendo de la demanda para un conjunto de recursos.

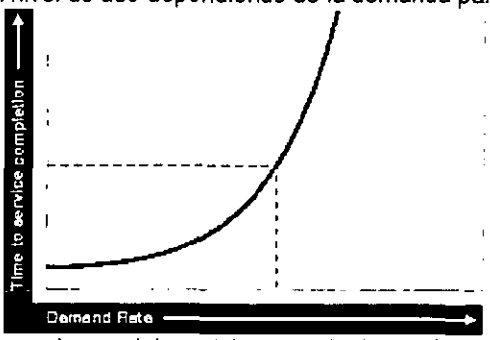


Fig.1 Tiempo de uso del servicio contra la demanda de recursos

Para manejar esta situación se tienen dos opciones:

- Se puede limitar la demanda para mantener tiempos de respuesta aceptable
- Agregar múltiples recursos, por ejemplo otro CPU o disco duro.

Para obtener el mejor rendimiento del sistema se deben conocer los cuatro componentes básicos de una máquina en su entorno de interacción con oracle y como puede afectar su rendimiento:

Memoria
Los cuellos de botella en la memoria ocurren cuando no existe suficiente memoria para dar cabida a las necesidades del sistema, cuando esto pasa, se da una excesiva paginación (movimientos de porciones de procesos al disco) y alternancia (transferencia de procesos enteros de memoria al disco para liberar memoria).

Entrada y salida del disco y controles
Los cuellos de botella en el disco y los controles ocurren cuando uno o más discos un controlador de disco excede el rango de entrada/salida recomendado.

CPU
Los problemas de CPU se dan cuando el sistema operativo o los programas de usuario están muy demandados sobre el CPU, esto es causado por demasiada paginación y alternancia o bien por un inadecuado número de procesadores en la máquina.

Red
Los problemas en la red se dan generalmente cuando la cantidad de tráfico sobre la red es demasiado grande o cuando ocurren colisiones sobre la red, los problemas de red se dan muy a menudo dentro de sistemas cliente-servidor o un entorno de sistema distribuido.

La memoria, el disco (I/O), CPU y problemas de red se deben minimizar. Cada uno de estos problemas afecta a los otros recursos también, de la misma manera, cualquier mejora en uno de estos recursos puede causar un mejor rendimiento de los otros, por ejemplo:

- El CPU puede necesitar más tiempo para manejar las operaciones de paginación y alternancia que se den debido a que la memora es escasa. Para esto debería adquirirse más memoria
- Si se tiene un cuello de botella en el sistema de entrada/salida del disco, se debe ser capaz de usar la memoria para almacenar más datos, así se evita tener lecturas de datos desde el disco.
- Si se tienen problemas de tráfico en la red en un entorno de cliente-servidor se debe mejorar el uso de la memoria, el CPU y la entrada/salida del disco, ya sea sobre el cliente o sobre el servidor para minimizar la transferencia de datos a través de la red.

En la afinación de los recursos, la meta es usar todos los recursos como sea posible. Algunas personas piensan que existe un buen rendimiento cuando la memoria es abundante y los problemas del disco y el CPU están cerca de cero y los problemas de tráfico de red son bajos. Pero más bien esto parecería una característica de un sistema no usado o que los usuarios han decidido no usar. Entonces, lo que realmente constituiría una verdadera afinación de sistema consideraría los siguientes puntos:

- Que la memoria sea usada lo más cerca al 100%
- La carga de los discos se extienda uniformemente por los dispositivos y todos los discos estén operando dentro de los rangos de entrada/salida recomendados
- Que el CPU esté siendo usado lo más cerca al 100% durante las horas pico sin tener programas que estén en espera de tiempo de CPU.
- Que el tráfico de la red esté por debajo del máximo recomendado sin colisiones.

Efectos de demanda excesiva

La demanda excesiva puede traer como resultado un gran aumento en el tiempo de respuesta, si existe la posibilidad de que la demanda exceda los niveles permisibles, entonces se debe limitar esta demanda

Los problemas de rendimiento pueden ser relevados a través de ajustar las unidades de consumo, algunos problemas pueden ser resuelto usando menos recursos por transacción o por reducción de los tiempos de servicio. Alternativamente se puede reducir el número de entradas y salidas del disco por transacción. Otros problemas pueden ser abatidos a través de la redistribución del trabajo y por reasignación de recursos. Si se usan múltiples CPUs en lugar de uno solo, se tendrán entonces múltiples recursos que podrían usarse.

Por ejemplo, si los tiempos más ocupados del sistema están entre las 9:00am y la 1:00pm se puede generar un plan para correr procesos en segundo plano después de las 2:30 cuando haya más capacidad disponible, de esta forma se puede manejar de una mejor manera la demanda. Alternativamente se generan márgenes para el manejo de los tiempos pico. **Figura 2**.

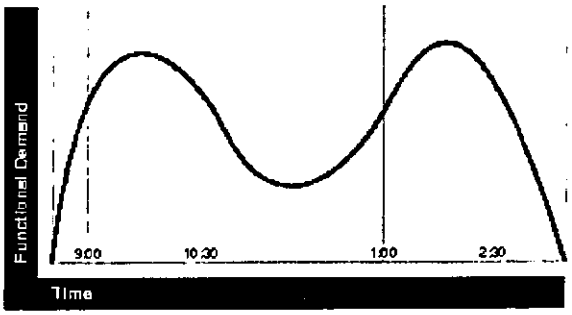


Fig. 2 Ajuste de la capacidad y la demanda funcional

Todas las personas relacionadas con el sistema tienen el papel de afinar los procesos. Cuando la gente comunica y documenta las características del sistema, la afinación puede llegar a ser significativamente fácil y rápida

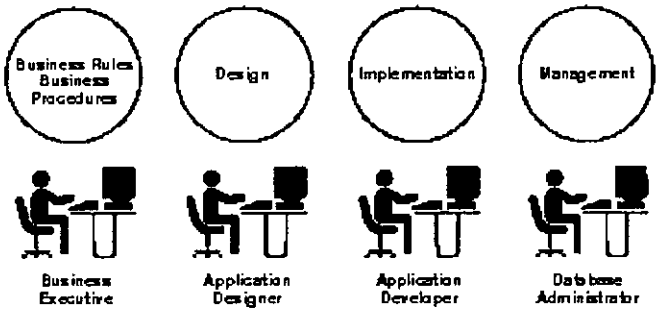


Fig. 3 Personas encargadas de la afinación del sistema

- Los ejecutivos del negocio deben definir y reexaminar las reglas del negocio y procedimientos para proveer y aclarar modelos adecuados de diseño de aplicación. Ellos son los responsables de identificar los tipos de información que son de interés para la compañía
- Los diseñadores de la aplicación deben comunicar el diseño del sistema de tal suerte que cualquier persona puede entender el flujo de los datos en la aplicación
- Los desarrolladores deben comunicar sobre las estrategias de implementación y elegir los módulos y comandos SQL que puedan ejecutarse más rápidamente y de fácil identificación durante la afinación de comandos.
- Los administradores de base de datos (DBAs) deben monitorear y documentar cuidadosamente las actividades del sistema de tal forma que puedan identificar y corregir ejecuciones inusuales del sistema.
- Los administradores de hardware y software deben documentar y comunicar la configuración del sistema de tal suerte que cualquier persona pueda diseñar y administrar el sistema fácilmente.

Las decisiones tomadas durante el diseño y desarrollo de la aplicación tienen en mayor impacto en el rendimiento. Una vez que la aplicación es liberada el DBA generalmente es el primer responsable de la afinación con las limitaciones obvias del diseño.

Cuando se diseña o mantiene un sistema, se deberían especificar metas de rendimiento de tal forma que se sepa cuando afinar. Se puede tener un desgaste significativo de tiempo sin una ganancia significativa si se intenta alterar la inicialización de parámetros o comandos SQL sin metas específicas.

Cuando se diseña un sistema se debe tener una meta bien específica: por ejemplo, una entrada tiene un tiempo de respuesta menor a tres segundos. Si la aplicación no reúne las metas, se generará un cuello de botella debido por ejemplo a contención, entonces, una vez que se determine la causa se debe corregir la acción. Durante la fase de desarrollo se debe hacer una prueba de la aplicación para determinar si se reúnen las metas del rendimiento antes de liberar la aplicación

Una vez que se han determinado los problemas, se pueden sacrificar algunas áreas para generar los resultados deseados. Por ejemplo, si tenemos un problema de I/O, podría ser necesario comprar mas memoria o más discos, si la compra no es posible se debe manejar un límite de concurrencia en el sistema y así generar el rendimiento deseado. Sin embargo si se tienen claramente definidas las metas del rendimiento, la decisión de inversión en el rendimiento es simple debido a que se tienen perfectamente identificadas las áreas más importantes.

Los desarrolladores de aplicaciones y los administradores deben ser muy cuidadosos para brindar el rendimiento más apropiado con respecto a las expectaciones de los usuarios. Cuando se tiene un sistema operativo muy complicado, el tiempo de respuesta puede ser muy lento aun cuando este ejecute una operación sencilla.

Evaluación del Rendimiento

Con metas claras y muy bien definidas, se puede determinar cuando la afinación del rendimiento ha sido exitosa. El éxito depende de los objetivos que se han establecido con la comunidad de usuarios, y la habilidad de ejercer acciones correctivas que superen los errores.

El DBA es responsable de resolver problemas de rendimiento y debe tener una visión amplia de todos los factores que determinan el tiempo de respuesta. El DBA debe monitorear la red, el disco, CPU, etc. a fin de localizar la fuente del problema en lugar de asumir que todos los problemas son generados por la base de datos.

El monitoreo del rendimiento del sistema ayuda a mantener un buen sistema afinado, el guardar una historia del rendimiento de las aplicaciones sirve como punto de comparación muy útil. Con datos sobre los consumos actuales de los recursos, se pueden hacer estudios y así predecir los requerimientos para futuros volúmenes de cargas.

La **figura 4** nuestra muchos factores envueltos en el rendimiento de una base de datos oracle que se deben considerar para un buen diseño de una aplicación, la afinación de esos factores es efectiva solamente después de que se afinan los procesos del negocio

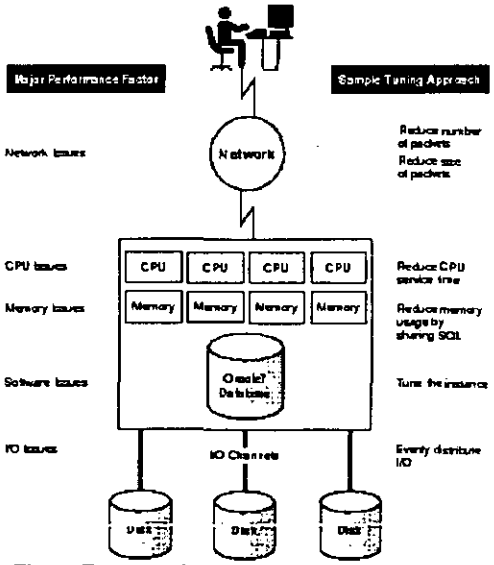


Fig. 4 Factores importantes del rendimiento

Los problemas de rendimiento tienden a ser interconectados más que aislados. La siguiente tabla muestra los factores de rendimiento en un sistema existente y las áreas en las cuales los síntomas de crisis pueden aparecer.

Areas de afinación Oracle	**Recursos**				
Aplicación	**CPU**	**Memoria**	**I/O**	**Red**	**Software**
Diseño/Arquitectura	X	X	X	X	X
DML SQL	X	X	X	X	X
Query SQL	X	X	X	X	X
Cliente/Servidor	X			X	
Instancia					
Buffer Cache	X	X	X		
Shared Pool	X	X			
Sort Area	X	X	X		
Estructura física de datos/DB File I/O	X		X		
Log File I/O		X	X		
Archiver I/O	X		X		
Segmentos Rollback			X		X
Locking	X	X	X		X
Backups	X		X	X	X
Sistema operativo					
Manejo de Memoria	X	X	X		
Manejo I/O	X	X	X		
Manejo de Procesos	X	X			
Manejo de Red		X		X	

Métodos de Afinación

La metodología es la llave para hacer una afinación del rendimiento con éxito. Existen diferentes estrategias de afinación que ofrecen cada una disminución en costos, esto es muy importante, se sabe que una buena estrategia trae un máximo de ganancias. Los sistemas tienen diferentes propósitos tales como transacción de procesos en línea y sistemas de soporte.

Para iniciar con el método partiremos de el hecho de que los ejecutivos del negocio deben colaborar con los diseñadores de la aplicación para establecer metas de rendimiento justificables. Durante la fase de diseño y desarrollo los diseñadores de la aplicación pueden determinar cual combinación de recursos de sistema y que características de oracle están disponibles para cubrir estas necesidades.

A través del un buen diseño de un sistema, se pueden minimizar los costos eventuales y la frustración. La **figura 5** muestra el costo relativo de la afinación durante el ciclo de vida de una aplicación

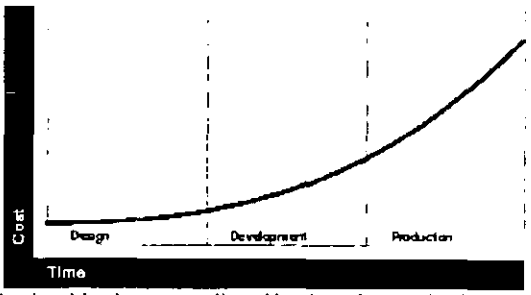


Fig. 5 Costo de Afinación durante el ciclo de vida de una aplicación. Las fases de desarrollo de un sistema en gran escala son el diseño, el desarrollo y la puesta en producción, en la fase de producción es más costoso el proceso de afinación.

Para complementar la figura anterior se pueden mostrar también los beneficios de afinar una aplicación durante del curso de vida, es decir a la par del crecimiento del sistema, y se puede observar que el costo es inversamente proporcional al anterior. Como se puede observar en las representaciones, el mejor tiempo para afinar es durante la fase de diseño ya que se obtiene el máximo beneficio al más bajo costo.

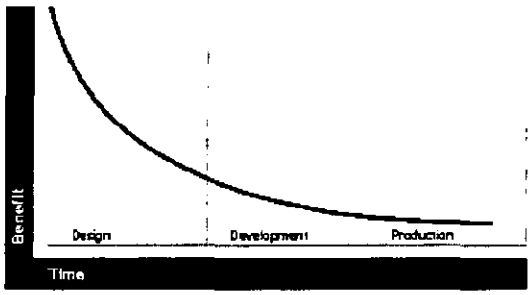


Fig. 6

Sin embargo, muchas personas piensan que la afinación se debe comenzar en el momento que los usuarios comienzan a quejarse sobre los tiempos de respuesta. Empero, este tiempo es el peor para usar estrategias de afinación eficaces. En este momento si no se decide a rediseñar la aplicación se pueden quizá tener mejoras de rendimiento solamente de asignación de memoria y del sistema I/O.

No obstante, es posible trabajar reactivamente para afinar un sistema existente. Para hacer esto, se debe tener un método y trabajar en primera instancia el cuello de botella. Una meta común es hacer que oracle corra rápidamente en una plataforma dada. Se puede dar también que el oracle server y el sistema operativo trabajen bien para obtener mejores rendimientos se pueden adicionar recursos Sin embargo solamente se pueden tener todas las ventajas de las múltiples características que oracle provee para mejorar el rendimiento cuando se realiza adecuadamente el diseño del sistema.

PASOS DEL METODO DE AFINACION

El método recomendado para afinar una base de datos oracle contiene pasos que prioritizan la disminución de costos: a continuación se enumeran para posteriormente ir hablando de cada uno de ellos:

- Afinación de las Reglas del Negocio
- Afinar el diseño de datos
- Afinar el diseño de la aplicación
- Afinar el entorno de la base de datos
- Afinar los SQL
- Afinar el CPU y arquitectura del sistema
- Afinar las asignaciones de memoria
- Afinación de los dispositivos de entrada y salida (I/O)
- Afinar la contención de Recursos

1. AFINACIÓN DE LAS REGLAS DEL NEGOCIO

Para un rendimiento óptimo, se tiene que adaptar el sistema a las reglas del negocio. Esto quiere decir que se debe hacer un análisis muy minucioso en el diseño del sistema. Las configuraciones que podrían considerarse en este nivel serían tales como y cuando usar o no servidores multitarea, de esta manera la planeación asegura que los requerimientos de rendimiento del sistema correspondan directamente con necesidades concretas de reglas de negocios.

Los problemas de rendimiento encontrados por el DBA pueden haber sido causados por problemas en el diseño e implementación o debido al manejo de reglas del negocio inapropiadas. Generalmente se llega a problemas graves cuando se diseñan reglas del negocio muy abstractas y se tratan de convertir en funciones de una aplicación, de esta forma los diseñadores tienen muchos problemas al elegir cuál debe ser la más apropiada implementación. Las reglas del negocio deberían ser consistentes y realistas con respecto a las expectativas de los usuarios actuales, el tiempo de respuesta y la cantidad de registros almacenados en línea que el sistema puede soportar.

2. AFINAR EL DISEÑO DE LOS DATOS

En la fase de diseño de datos se debe determinar que datos son verdaderamente necesarios para la aplicación. Es necesario considerar también cuales relaciones son muy importantes y cuales son sus atributos. Y no menos importante es estructurar la información de tal suerte que cumpla de la mejor manera con las metas del rendimiento y ejecución.

El proceso de diseño de la base de datos debe sufrir un proceso de normalización en el cual se asegure que sean analizados los datos y de esta manera evitar redundancia. Una vez que los datos son cuidadosamente normalizados podría ser necesario desnormalizar por razones de rendimiento. Se puede necesitar por ejemplo que la base de datos mantenga valores sumarizados frecuentemente; en lugar de forzar a una aplicación a recalcular el precio total de toda una orden dada cada que esta es accesada, se podría decidir incluir el valor total de cada orden en la base de datos. Se podrían generar índices de llaves primarias o foráneas para accesar a esa información rápidamente.

Otra consideración en el diseño de datos es la anulación de la contención sobre los datos. Puede considerarse una base de datos con el tamaño de un terabyte sobre la cual miles de usuarios accesan solamente el 0.5% de los datos, esto puede causar problemas en el rendimiento. La contención comienza cuando se accesa remotamente a los datos y la cantidad de contención está determinada por la escalabilidad.

3. AFINAR EL DISEÑO DE LA APLICACIÓN

Los ejecutivos del negocio y los diseñadores de la aplicación necesitan trasladar las metas de las necesidades del negocio en un diseño de sistema efectivo, los procesos del negocio concernientes a una aplicación en particular dentro del sistema o como una parte en particular de una aplicación.

En este nivel también se puede considerar una configuración individual de procesos, por ejemplo algunos usuarios de PC pueden accesar el sistema central usando accesos remotos, mientras que otros usuarios se conectan directamente, estos aunque corren en el mismo sistema, su arquitectura es diferente, estos pueden tener diferentes servicio de correo o diferentes versiones de la aplicación

4. AFINAR EL ENTORNO DE LA BASE DE DATOS

Después que una aplicación ha sido diseñada se puede planear el entorno de la base de datos. Un entorno de afinación es una parte esencial de una base de datos; si el entorno de la base de datos no se afina esto puede tener una influencia negativa impactando sobre toda transacción hecha por los usuarios. El entorno de la base de datos está dado por los parámetros manejados en el init.ora que determinarán en primera instancia la eficiencia de esta. Los parámetros más importantes se describen a continuación.

DB_BLOCK_SIZE

El parámetro DB_BLOCK_SIZE se fija cuando la base de datos es creada y determina el tamaño de cada bloque dentro de la base de datos, no se puede cambiar el tamaño de un bloque de una base de datos existente, el único método disponible para incrementar el tamaño del bloque es ejecutando un export completo de la base de datos y recrear la base de datos con un DB_BLOCK_SIZE con un valor mayor y hacer entonces un import de la base de datos. La recreación completa de la base de datos es generalmente una operación muy costosa en términos de tiempo y forza también a aumentar el nivel de crisis, debido a ello, se debe asegurar fijar un tamaño apropiado para este parámetro cuando la base de datos es creada por primera vez.

En muchos entornos el valor default para el este parámetro es de 2048 bytes (2kb), si el sistema operativo lo permite se debe incrementar ese valor a 4kb, 8kb u otro mayor. La ganancia de rendimiento obtenida por usar un valor grande para este parámetro es muy significativo para las aplicaciones en batch, generalmente cada que se incrementa su valor se debería reducir el tiempo requerido por operaciones batch por lo menos en un 40%, pero como el tamaño del bloque se incrementa, los requerimientos de memoria también se incrementan.

DB_BLOCK_BUFFERS

El parámetro DB_BLOCK_BUFFERS fija su tamaño en los database blocks de los data block buffer cache en el SGA . El tamaño del data block buffer cache es el mayor número de memoria disponible para compartir datos que ya estén en memoria.

El archivo init.ora provee tres posibles valores para el parámetro DB_BLOCK_BUFFERS, ("SMALL", "MEDIUM" y "LARGE"), dependiendo del número de usuarios, se usa el MEDIUM cuando existen de seis a diez usuarios concurrentes en la base de datos, si se tienen menos de seis debe usarse SMALL y si existen más de diez se debe fijar este parámetro a LARGE. Una vez que la base de datos ha sido puesta en uso, se debe monitorear el impacto para saber cuando se necesita incrementar este parámetro. Si se tienen menos usuarios, los valores SMALL o MEDIUM son apropiados, de otra forma se necesitará usar el valor LARGE. Generalmente el tamaño del data block buffer cache debería estar entre 1 y 2 porciento del tamaño de la base de datos.

SHARED_POOL_SIZE

El parámetro del SHARED_POOL_SIZE fija el tamaño en bytes de shared pool en el SGA; la shered pool almacena datos para el library caché (para sistemas que usan servidor multitarea) y datos específicos de la sesión. Si la aplicación usa paquetes y otros objetos procedurales el tamaño del shared pool puede exceder el del data block buffer caché. Si se usa un servidor multitarea se podría necesitar el doble del tamaño del shared pool. El archivo init.ora provee tres posibles valores para el SHARED_POOL_SIZE por default (SMALL, MEDIUM Y LARGE). Si se tienen muchos usuarios se debería incrementar el tamaño del parámetro SHARED_POOL_SIZE cada que se incremente el parámetro DB_BLOCK_BUFFERS.

DB_WRITERS

Una vez que el tamaño de los bloques en las áreas de memoria están propiamente establecidos, se debe afinar el camino a través del cual oracle escribe datos desde la memoria. Si el sistema operativo lo permite se deberían usar múltiples procesos DBWR usando un valor mayor que 1 en el parámetro DB_WRITERS, si se inicia con más de un proceso DBWR se puede reducir la contención de los bloques dentro del data block buffer cache. Se solo se tiene un solo proceso DBWR disponible, este puede

generar entonces un cuello de botella durante las operaciones de entrada/salida (I/O) siempre que los datos estén distribuidos en diferentes máquinas.

Se pueden usar diferentes factores para ayudar a estimar el número de DBWR que se podrían necesitar. Primeramente, se puede determinar el número de conexiones concurrentes. Se puede consultar para esto la tabla V$LICENSE para verificar el número máximo de conexiones concurrentes desde que la base de datos fue iniciada. El número de procesos DBWR debe ser proporcional al número de transacciones y al número de usuarios que actualizan la base de datos simultáneamente. En general, se debe tener un proceso DBWR para cada 50 usuarios en línea (haciendo consulta o actualizando) y un proceso DBWR por cada dos procesos batch que actualizan la base de datos.

SORT_AREA_SIZE y SORT_AREA_RETAINED_SIZE

Una vez que se ha establecido el entorno que puede leer datos desde la memoria efectivamente, se necesitará entonces establecer el tamaño de las áreas de memoria usada para ordenamientos. Si las áreas de ordenamiento no son lo bastante grandes, oracle creará entonces segmentos temporales para usarlos durante las operaciones de ordenamiento; por lo tanto, si se puede ejecutar todo el ordenamiento en memoria, se puede también eliminar el costo de escribir datos en segmentos temporales.

El parámetro SORT_AREA_SIZE especifica la cantidad máxima en memoria en bytes que el usuario tiene disponible para ordenamientos. El parámetro SORT_AREA_RETAINED_SIZE muestra la cantidad máxima de memoria en bytes que puede ser usada por un ordenamiento en memoria.

ROLLBACK_SEGMENTS

Para soportar propiamente las transacciones dentro de la base de datos, se necesita tener bastantes segmentos de Rollback. Entre más transacciones concurrentes que se den en la base de datos se necesitan más segmentos de Rollback (sin embargo para el disco donde se almacenan esos segmentos de Rollback se puede crear un cuello de botella durante los procesos de transacción).

Se debe tener un conjunto de segmentos de Rollback específicamente diseñado para soportar la ejecución de transacciones más frecuentes dentro de una aplicación. Se deben tener por separado grupos de segmentos de Rollback que tengan un tamaño suficiente para soportar grandes transacciones. Para determinar si se tienen suficientes segmentos de Rollback se puede hacer una consulta (Query) sobre la tabla V$WAITSTAT, esta tabla lista los diferentes tipos de estadísticas de espera almacenadas. Si las estadísticas muestran un gran tiempo de espera y que además se incrementa continuamente, entonces se necesitará crear más segmentos de Rollback de tal surte que efectivamente soporte las transacciones en la base de datos.

Afinación de memoria

En forma ideal una base de datos realiza los menos accesos físico como le sea posible, usando las áreas del SGA para compartir datos entre los usuarios que ya fueron previamente leídos. El sistema operativo usa estrategias similares tales como almacenar los archivos en un buffer para mejorar los accesos (I/O) del sistema.

Cuando un archivo del sistema es usado el sistema operativo lee los datos dentro del buffer cache[1], entonces los datos son transferidos al oracle buffer cache en el SGA , esto es parte de un proceso de memoria dentro del sistema operativo. Tanto los procesos de memoria como el kernel tienen el mismo límite físico disponible de memoria y deben ser manejados como un todo. Dependiendo del hardware los arreglos del disco pueden proporcionar buffers adicionales para decrementar los tiempo de acceso, particularmente los de las aplicaciones que hacen lecturas intensivas.

Cuando se maneja la memoria para una aplicación de base de datos oracle, se debe primero manejar la memoria dentro de oracle. Si se crea un entorno estable para las áreas de memoria de oracle se debe reducir la probabilidad de contención de memoria a nivel de sistema operativo.

[1] Este ejemplo se refiere al sistema operativo UNIX y para ser específicos se refiere al UNIX buffer caché el cual es parte de la memoria del kernel

Contención de memoria

Para reducir la contención de diferentes áreas de memoria se necesitan manejar las ocurrencia de dos niveles de memoria virtual en el sistema operativo: el swapping (alternancia) en el cual procesos enteros son movidos de la memoria hacia el disco y paging (paginación) en la cual secciones seleccionadas de un proceso son leídas y/o escritas fuera de memoria.

Alternancia

Cuando se necesita memoria para servicios de procesos demandados por el sistema y la cantidad de memoria disponible esta por debajo del necesario, el sistema operativo tendrá que generar espacio disponible para intentar mover procesos enteros fuera de memoria hacia el disco, este espacio en el disco se le conoce como swap space. La operación de alternancia genera un espacio virtual de procesamiento para el sistema operativo. También se da en sentido inverso, los procesos que están en el swap space regresan a memoria para continuar siendo ejecutados.

El swap space es una área en el disco asignada por el sistema operativo para almacenar la memoria usada por procesos que han sido puestos fuera de la memoria; para oracle es recomendado que este espacio sea configurado al menos de dos a cuatro veces de la cantidad de memoria física.

Paginación

La mayoría de los sistema operativos proveen un nivel más fino de manejo de memoria en el cual solamente paginas seleccionadas (partes de memoria individual) de un proceso son leídas y/o escritas fuera de memoria. Este proceso es llamado paginación y generalmente es manejado por los algoritmos de el menos recientemente usado, y trabaja en forma similar que el método del SGA de oracle.

Memoria compartida

La memoria compartida es una estructura usada por múltiples procesos que comparten información. En muchos sistemas operativos, oracle está implementado con una arquitectura multitarea donde varios procesos accesan a los recursos compartidos y los datos por el SGA. Todos los procesos de oracle deben estar dentro del SGA. El tamaño del SGA se determina en la inicialización de la base de datos basándose en algunos parámetros.

El SGA reside en la memoria compartida del sistema operativo, debido a esto, la memoria compartida debe estar disponible en el sistema operativo cuando la base de datos es levantada. La memoria compartida esta disponible en segmentos y múltiples segmentos pueden ser usados por un proceso.

El propósito de la estructura de memoria de oracle es proveer el mejor acceso dentro de oracle. Si el área del SGA es grande y se comienza a dar una paginación excesiva debido a que no hay suficiente memoria para otros procesos, esto puede convertirse en un problema en lugar de un recurso.

Por ejemplo, si parte del SGA es puesta fuera de memoria y una lectura es hecha a un bloque que ha sido escrito en el swap space, entonces el sistema operativo tiene que hacer un acceso al disco para satisfacer el requerimiento de los datos. Configurado el SGA en forma adecuada los requerimientos son satisfacidos desde el buffer cache y la información se recupera en forma lógica y no de una lectura física.

Implementación de un arquitectura flexible y óptima

Una arquitectura flexible y óptima (OFA) es un conjunto de requerimientos y reglas que se definen en las guías de instalación donde se incluye estructuras de directorio uniformes, nombramientos estándares y guías para los datafiles. Un OFA es obligatorio para simplificar la administración del entorno de oracle. Los requerimientos de un OFA pueden ser considerados metas y objetivos para tener un entorno flexible y óptimo.

Los requerimientos incluyen:

- Organización de los archivos de sistema y nombramiento estándar que hacen el entorno escalable y fácil de administrar
- Distribución de archivos para un balance de accesos (I/O)
- Un apropiado aislamiento para múltiples versiones de entornos oracle (desarrollo, pruebas, producción, etc.) en el mismo servidor

Las reglas de OFA sirven como un ejemplo de organización de directorios y de archivos para bases de datos oracle, estas incluyen algunos puntos tales como:

- Reglas de organización de directorios
- Convenciones de nombramientos de directorios y archivos
- Organización de archivos
- Guías de nombramiento de Tablespaces

Las reglas de OFA intentan solo informar y no darlas como un hecho. El nombre exacto de archivos y directorios no es tan importante, solo para estandarizar y simplificar y no para causar confusiones.

El objetivo más importante relacionado al manejo de archivos de la base de datos es para garantizar que todos los datafiles tengan la misma estructura usando las convenciones de nombramiento, de tal suerte que si el nombre de una instancia se llama PROD la estructura de directorio usada por los datafiles se debería llamar:

/db01/oracle/ PROD
/db02/oracle/ PROD
/db03/oracle/ PROD
/db04/oralce/ PROD

La estructura del directorio de la instancia TEST debería ser:

/db01/oracle/ TEST
/db02/oracle/ TEST
/db03/oracle/ TEST
/db04/oralce/ TEST

Dentro de cada directorio el nombre de los archivos es consistente. Si cada instancia tiene un directorio separado no se necesita incluir el nombre de la instancia en los nombres de archivo, si se necesita más espacio para los archivos del que está disponible, se deben reasignar los archivos como sea necesario pero manteniendo el directorio con las reglas de OFA .

Usando un nombramiento estándar de archivos y directorios se simplifica la administración, espacialmente para procesos de respaldo.

Distribución de accesos (I/O)

Dentro de una base de datos se pueden mover los objetos más frecuentemente accesados dentro de tablespaces; pueden existir tablas separadas y sus indices relacionados; segmentos de Rollback separados. Fuera de la base de datos se necesita asegurar que la distribución de archivos soporte la distribución lógica dentro de la base de datos.

Por ejemplo, si se tienen por separado dos tablas muy usadas en sus propios tablespaces, pero puestas por los datafiles de esos tablespaces sobre el mismo disco entonces no se ha eliminado el potencial de contención de acceso entre las tablas. Para esto se deberían poner ambas tablas en el mismo tablespace, de esta manera se disminuye el potencial de generar un cuello de botella.

5. AFINAR LOS SQL

El lenguaje de consulta estructurado (SLQ) es la parte medular de Oracle. Este es un lenguaje flexible y se pueden usar diferentes comandos que generan los mismos resultados aun bajo diferentes construcciones pero solo uno de ellos es el más eficiente en una situación dada.

Un SQL eficiente es aquel en el que el resultado se obtiene en el menor tiempo posible sin hacer menor el rendimiento de los recursos del sistema. Para afinar querys primero se deben determinar los pasos para optimizar que se deberán seguir para resolver las consultas. Para hacer esto se puede usar el comando **explain plan** de oracle que despliega el plan de ejecución de un Query dado. Este comando evalúa los pasos que un Query recorre al ejecutarse y entonces pone un registro por cada paso en la tabla de nombre PLAN_TABLE. Los registros en el PLAN TABLE describirán las operaciones usadas en cada paso de la ejecución del Query y las relaciones entre estos pasos.

Para usar el explain plan primero se necesita crear la tabla de PLAN_TABLE en el esquema que se usará[2]. Oracle provee un script para crear el PLAN_TABLE este se llama utlxplan.SQL y generalmente se encuentra en el directorio de /RDBMS/ADMIN sobre el oracle home.

La estructura del PLAN_TABLE es la siguiente:

```
Name                           Type
------------------------------ ----
STATEMENT_ID                   VARCHAR2(30)
TIMESTAMP                      DATE
REMARKS                        VARCHAR2(80)
OPERATION                      VARCHAR2(30)
OPTIONS                        VARCHAR2(30)
OBJECT_NODE                    VARCHAR2(128)
OBJECT_OWNER                   VARCHAR2(30)
OBJECT_NAME                    VARCHAR2(30)
OBJECT_INSTANCE                NUMBER(38)
OBJECT_TYPE                    VARCHAR2(30)
OPTIMIZER                      VARCHAR2(255)
SEARCH_COLUMNS                 NUMBER(38)
ID                             NUMBER(38)
PARENT_ID                      NUMBER(38)
POSITION                       NUMBER(38)
COST                           NUMBER(38)
CARDINALITY                    NUMBER(38)
BYTES                          NUMBER(38)
```

Una ves que la tabla PLAN_TABLE ha sido creada en el esquema se puede dar inicio al uso del explain plan para múltiples querys, las columnas del plan table se describen en la siguiente tabla.

Columna	Descripción
Statement_ID	Esta columna almacena el nombre del Query para futuras referencias
Timestamp	Esta columna almacena cuándo se emitió el comando explain plan
Remarks	La columna de Remark puede agregar un comentario a registros existentes en el PLAN_TABLE a través de un comando update
Operation	La operación que se ejecutó en ese paso
Options	La opción usada por la operación, tal como UNIQUE SCAN OR RANGE SCAN por una operación de INDEX
Object_Node	El link de base de datos usado para referenciar un objeto

[2] Generalmente el esquema es el mismo donde están las tablas en donde se hará la consulta

Columna	Descripción
Object_Owner	El propietario del objeto
Object_Name	El nombre del objeto
Object_instance	La posición ordinaria del objeto
Obect_Type	Un atributo de objeto tal como UNIQUE para indices
Optimizer	El modo de optimización usada (tal como FIRST_ROWS o RULE)
Search_Columns	No usado actualmente
ID	Un número asignado a cada paso en el plan de ejecución; con el parent_id establece una estructura jerárquica.
Parent_ID	El id del paso que es el padre del paso actual en la estructura jerárquica del plan de ejecución
Position	El orden de procesamiento para pasos que tienen el mismo parent_id. En el primer registro generado por el explain plan, esta columna contiene una estimación de optimización de los comandos si la base de datos esta configurada en modo de costos.
Cost	El costo relativo del paso si la optimización es basada en costos; esta columna solo está disponible en la versión de oracle 7.3
Cardinality	Es el número esperado de registros retornados desde la operación
Bytes	Es el tamaño de cada registro retornado

Para hacer una afinación de procesos simple, siempre se debe usar el mismo Statement_id y borrar los registros de cada plan de ejecución antes de usar el comando explain plan una segunda vez.

A continuación se muestra un ejemplo de plan de ejecución con el comando explain plan en donde el Query que se muestra no correrá durante la ejecución del comando, solamente se usa para generar los pasos del plan de ejecución insertándolos en la tabla de PLAN_TABLE

```
Explain plan
Set statement_id = 'test'
For
Select Name, City, Sate
From   company
Where  City = 'Roanoke'
And    State = 'VA';
```

Después de correr este comando se insertan un conjunto de registros en la tabla de PLAN_TABLE. Se puede consultar esta tabla usando el siguiente Query. Los resultados de este Query mostrarán las operaciones ejecutadas cada paso y la relación padre-hijo entre los pasos del plan de ejecución.

```
Select  LPAD(' ',2*Level)||Operation||' '||Options||' '||Object_name Q_PLAN
From plan_table
Where Statement_ID = 'test'
Connect  by prior ID = Parent_ID
Start with ID = 1;
```

Que da como resultado :

```
Q_PLAN
----------------------------------------
TABLE ACCES COMPANY BY ROWID
  AND EQUAL
    INDEX RANGE SCAN COMPANY$CITY
    INDEX RANGE SCAN COMPANY$STATE
```

Si se usa una versión de oracle 7.3 o superior se puede tener el plan de ejecución generado automáticamente para todas las transacciones que se ejecuten desde SQL*Plus. El comando SET AUTOTRACE ON causa que cada Query después de ser

ejecutado y despliega tanto el plan de ejecución e información del trace que se refiere acerca de los procesos envueltos para resolver el Query.

Interpretación de las operaciones

El plan de ejecución muestra en forma de lista la relación padre-hijo entre tres operaciones: TABLE ACCES BY ROWID, AND-EQUAL, e INDEX RANGE SCAN. Se procesa el plan de ejecución del nivel inferior al superior, así, las dos operaciones de INDEX RANGE SCAN pueden proveer datos para la operación AND-EQUAL. Esta última operación es solamente para proveer información para la operación TABLE ACCES BY ROWID.

Operaciones

Para interpretar un plan de ejecución y evaluar correctamente las opciones de afinación de un Query se necesita primero entender las diferencias entre las operaciones disponibles en la base de datos.

Las operaciones están clasificadas como operaciones de registro u operaciones de grupo, las diferencias entre ellas se muestran en la siguiente lista:

Operaciones de registro	***Operaciones de grupo***
Ejecutadas sobre un registro a la vez	Ejecutadas sobre un conjunto de registros a la vez
Pueden ser ejecutadas en un estado de FETCH, si estas no están envueltas en una operación de grupo	Ejecutadas en un estado de EXECUTE cuando un cursor está abierto
El usuario puede ver el primer resultado antes que el último registro sea recuperado	El usuario no puede ver el primer resultado mientras todos los registros sean recuperados y procesados
Ejemplo: un full database scan	Ejemplo: Un full table scan con una cláusula group by

A continuación se muestra un conjunto de operaciones en orden alfabético y su respectiva descripción

AND-EQUAL

Esta operación une listas de valores almacenadas retornadas por índices. Esta operación retorna la lista de valores que es común a ambas listas (tales como RowIDs que son encontrados en dos distintos índices). AND-EQUAL es usado para unir indices y range scans de índices únicos.

Ejemplo:

```
Select Name, City, Sate
From   company
Where  City = 'Roanoke'
And    State = 'VA';
```

El Query muestra en la lista de precedencia basándose en dos criterios en la cláusula Where, el criterio puede usar el índice COMPANY$CITY sobre la columna de City y el índice COMPANY$STATE sobre la columna de State para obtener RowIDs de los registros que sean retornados. Desde que la columna es requerida por el Query, la tabla COMPANY tendrá que ser accesada y los valores de otras columnas están ya disponibles a través de una búsqueda por índice.

El plan

```
TABLE ACCES COMPANY BY ROWID
  AND EQUAL
    INDEX RANGE SCAN COMPANY$CITY
    INDEX RANGE SCAN COMPANY$STATE
```

El plan muestra que los dos índices no únicos (sobre la columna city o sobre la columna State) son examinados y emparejados por los criterios de la cláusula Where. Los RowIDs de cada índice son puestos en listas almacenadas. La operación AND-EQUAL une las dos listas y genera una sola lista de RowIDs que estaban en ambas listas. Si un RowID existe solamente en uno de los índices, este no será retornado por la operación AND-EQUAL. Desde que la cláusula Where contiene un operador AND la operación AND-EQUAL es usada para evitar retornar RowIDs que no se encuentren en ambos índices. Los RowIDs son usados entonces para accesar los registros en la tabla COMPANY que satisfagan los criterios de la cláusula Where y para que el Query sea completado.

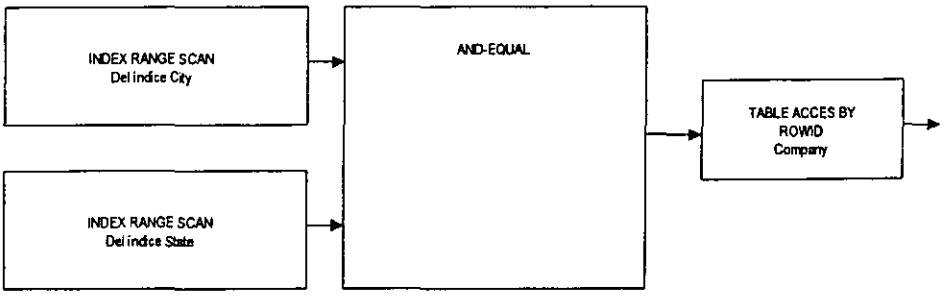


Fig. 7 Ejemplo de ejecución de AND-EQUAL

CONCATENATION

Esta operación se genera a nivel de registro, se genera como un UNION ALL (una unión sin eliminar los valores duplicados) por ejemplo:

```
Select Name, City, Sate
From   company
Where  State = 'TX'
And    City in ('Houston','Austin','Dallas');
```

Como el Query usa para columna City una cláusula IN y esta funcionalmente se comporta como un OR se puede reescribir el Query de la siguiente forma:

```
Select Name, City, Sate From company
Where  State = 'TX'
And   (City = 'Houston'
OR     City ='Austin'
OR     City ='Dallas');
```

Aun podemos llevar esta expresión un paso más, poniendo la validación de state en cada una de las cláusulas OR quedando de esta manera:

```
Select Name, City, Sate
From   company
Where (State = 'TX' AND City ='Houston')
OR    (State = 'TX' AND City ='Austin')
OR    (State = 'TX' AND City ='Dallas');
```

Este Query generaría un plan como el que se describe a continuación:

```
CONCATENATION
TABLE ACCES BY ROWID COMPANY
  AND EQUAL
    INDEX RANGE SCAN COMPANY$CITY
    INDEX RANGE SCAN COMPANY$STATE
TABLE ACCES BY ROWID COMPANY
  AND EQUAL
```

```
    INDEX RANGE SCAN COMPANY$CITY
    INDEX RANGE SCAN COMPANY$STATE
TABLE ACCES BY ROWID COMPANY
  AND EQUAL
    INDEX RANGE SCAN COMPANY$CITY
    INDEX RANGE SCAN COMPANY$STATE
```

El plan muestra que el Query es ejecutado como si la cláusula IN hubiera sido reescrita como una cláusula OR y como si el otro criterio de validación hubiera sido puesto dentro de la cláusula OR. Dentro de cada cláusula OR se ejecuta una operación AND-EQUAL para unir las listas de RowIDs retornada por cada index scan. Los RowIDs retornados por las operaciones AND-EQUAL son usados entonces para seleccionar las columnas requeridas de la tabla COMPANY via operación TABLE ACCES BY ROWID. Los registros que resultan de cada parte del Query son concatenados para generar la salida al usuario.

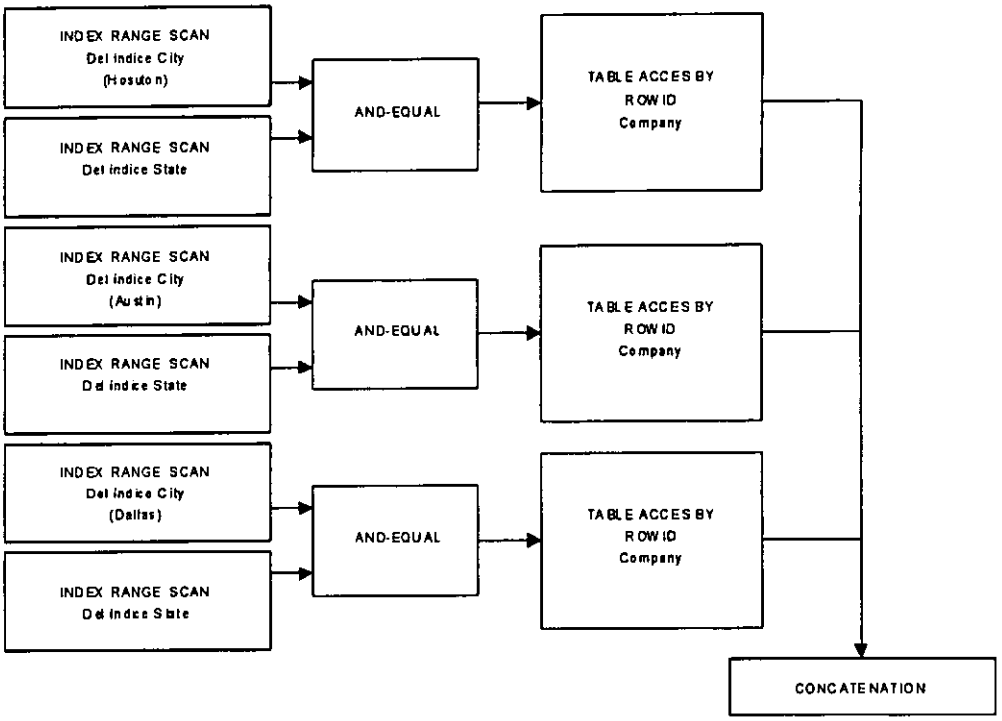


Fig. 8 Ejemplo de plan de ejecución de Concatenation

CONECT BY

CONNECT BY es un join de una tabla consigo misma en una forma jerárquica un ejemplo de este tipo de querys es el siguiente, en donde el resultado seria precisamente los nombres de las compañías en forma descendente desde el padre hasta los hijos tomando en consideración la jerarquía:

```
Select Company_ID ,Name
From   company
Where  State = 'VA'
Connect by prior Parent_company_ID = Company_ID
Start with Company_ID = 1;
```

Este Query generaría un plan de ejecución como el siguiente:

```
FILTER
CONNECT BY
 INDEX UNIQUE SCAN COMPANY_PK
 TABLE ACCES BY ROWID COMPANY
 TABLE ACCES BY ROWID COMPANY
  INDEX RANGE SCAN COMPANY$PARENT
```

El plan muestra que primero se usa el indices COMPANY_PK para encontrar el nodo raiz (Company_ID = 1), entonces el indices sobre la columna de Parent_copmpany_ID es usado para proveer los valores requeridos por la columna Company_ID en forma iterativa (como si fueran dos tablas separadas usando la operación TABLE ACCES BY ROWID). Después que la jerarquia es completada se hace una operación de FILTER.

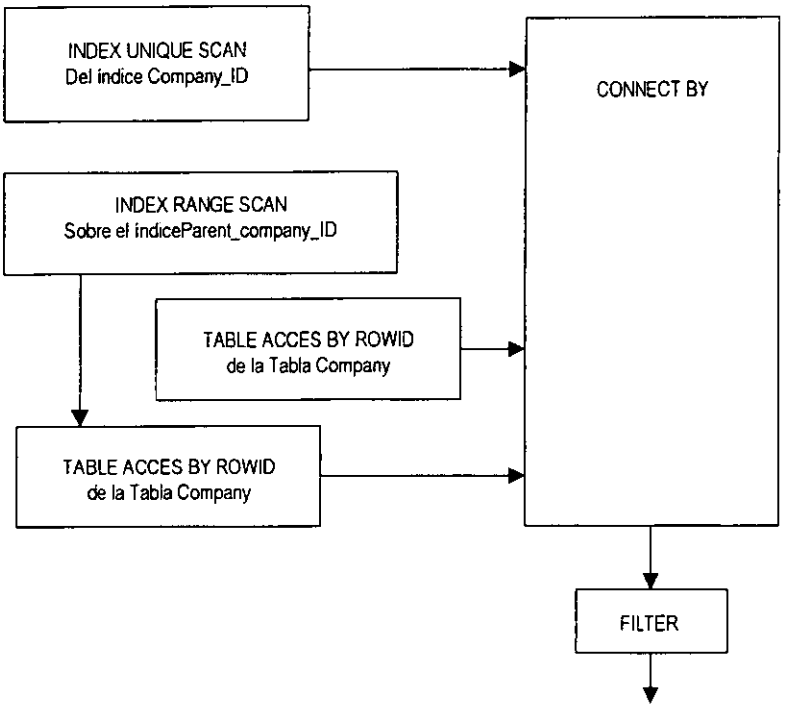


Fig. 9 Ejemplo de connect by

HASH JOIN

El HASH JOIN une tablas a través de crear en memoria una copia de la tabla y entonces hacer la unión con la segunda tabla. En el siguiente Query las tablas COMPANY y SALES son unidas basadas en una columna común a ambas (Company_ID)

```
Select COMPANY.Name
From   COMPANY, SALES
Where  COMPANY.Company_ID = SALES.Company_ID
And    SALES.Period_ID = 3
And    SALES.Sales_Total > 1000;
```

Su plan de ejecución es el siguiente:

```
HASH JOIN
 TABLE ACCES FULL SALES
 TABLE ACCES FULL COMPANY
```

El plan muestra que la tabla SALES es usada como pivote por lo que se hace una copia en memoria desde donde va a ser leída. Oracle usará la función de HASH para comparar los valores en la tabla COMPANY sobre los registros que han sido leídos de la memoria.

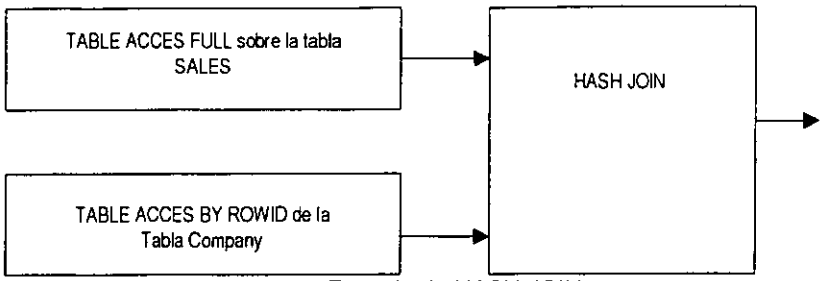


Fig. 10 Ejemplo de HASH JOIN

INDEX RANGE SCAN

INDEX RANGE SCAN selecciona un conjunto de valores desde un índice, el índice puede ser único o no único. Los valores encontrados son usados cuando alguna de las siguientes condiciones es dada:

- Un operador es usado (Tal como < ó >)
- La cláusula BETWEEN es usada
- Se busca un patrón de caracteres con el operador %
- Se usa solo como parte de un índice concatenado (por ejemplo un índice basado en dos columnas de una tabla)

El acceso al rango de valores dentro de un índice inicia con el primer registro que es incluido en el rango. Después que el primer registro ha sido localizado se hace un barrido horizontal sobre los bloques del índice hasta que el último registro es encontrado.

La eficiencia de un INDEX RANGE SCAN esta directamente relacionada a dos factores: el número de llaves en el rango de selección (entre más valores más grande es la búsqueda) y la condición del índice (Entre más fragmentado esté el índice, más grande es la búsqueda)

```
Select Name, City, State
From   COMPANY
Where  City > 'Roanoke';
```

Este Query genera el siguiente plan

```
TABLE ACCES BY ROWID COMPANY
 INDEX RANGE SCAN COMPANY$CITY
```

El plan muestra que el índice sobre la columna City es usado para encontrar Row_Ids en la tabla COMPANY que satisface la condición de la cláusula Where.

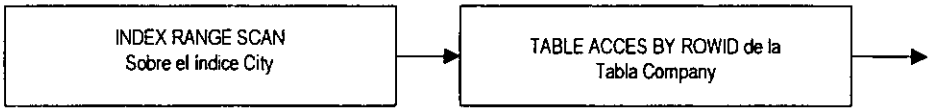


Fig. 11 Ejemplo de INDEX RANGE SCAN

INDEX UNIQUE SCAN

Esta operación selecciona valores únicos de un índice único, es el método más eficiente de selección de registros conocido. Podemos ejemplificar con el siguiente Query.

```
Select Name, City, State
From   COMPANY
Where  Company_ID > 12345;
```

Plan:

```
TABLE ACCES BY ROWID COMPANY
 INDEX UNIQUE SCAN COMPANY_PK
```

El Query usa la columna Company_ID como único criterio en la cláusula Where. Como Company_ID es la llave primaria de la tabla COMPANY, esta columna tiene un índice único asociado llamado COMPANY_PK. Durante el Query, el índice COMPANY_PK es barrido para buscar el valor ('12345'). Cuando el valor es encontrado, el RowID asociado es usado para consultar la tabla COMPANY.

Fig. 12 Ejemplo de INDEX UNIQUE SCAN

MERGE JOIN

MERGE JOIN une tablas a través de fusionar listas almacenadas de registros de cada tabla. Esta operación es efectiva para operaciones grandes pero puede ser también ineficiente para joins usados por procesos de transacción. MERGE JOIN es usado siempre y cuando oracle no pueda usar un índice mientras se ejecuta un join.

En el siguiente ejemplo se deshabilitaron deliberadamente los índices sumándole 0 a los valores numéricos durante el join, de esta forma se forza a que ocurra un merge join.

```
Select COMPANY.Name
From   COMPANY, SALES
Where  COMPANY.Company_ID+0 = SALES.Company_ID+0
And    SALES.Period_ID = 3
And    SALES.Sales_Total > 1000;
```

Plan

```
MERGE JOIN
 SORT JOIN
   TABLE ACCES FULL SALES
  SORT JOIN
   TABLE ACCES FULL COMPANY
```

Como se muestra en el plan, oracle ejecutará un búsqueda completa en la tabla (TABLE ACCES FULL) sobre cada tabla, ordena los resultados (usando la operación SORT JOIN) y une los resultados. El uso de merge joins indica que los índices están deshabilitados o no están disponibles por la sintaxis del Query.

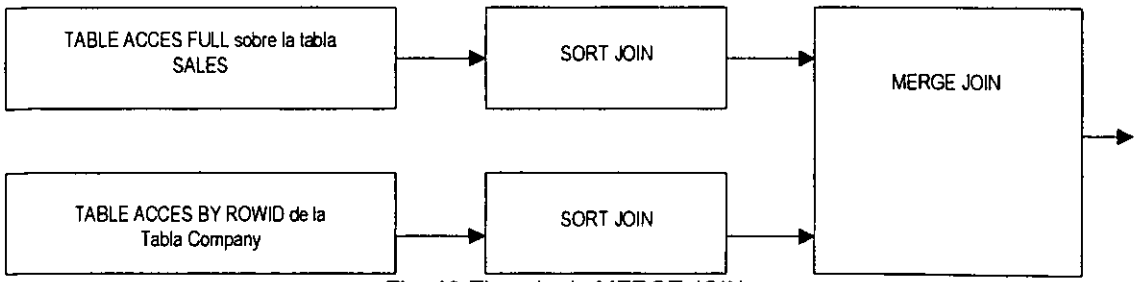


Fig. 13 Ejemplo de MERGE JOIN.

NESTED LOOPS

NESTED LOOPS une tablas accesando operaciones donde al menos una de las columnas que conforman el join está indexada. Ejemplo:

```
Select COMPANY.Name
From   COMPANY, SALES
Where  COMPANY.Company_ID = SALES.Company_ID
And    SALES.Period_ID = 3
And    SALES.Sales_Total > 1000;
```

Plan :

```
NESTED LOOPS
  TABLE ACCES FULL SALES
  TABLE ACCES BY ROWID COMPANY
   INDEX UNIQUE SCAN COMPANY_PK
```

El plan muestra que la tabla SALES es usada como pivote por el Query. Cuando se genera un join de tipo NESTED LOOPS, siempre una tabla es usada como pivote para manejar el Query, así para cada ocurrencia de Company_ID en la tabla SALES, el índice sobre Company_ID sobre la tabla COMPANY será checado para ver si existe un valor correspondiente. Si existe ese valor, el registro es retornado al usuario a través de una operación NESTED LOOPS.

Existen diferentes observaciones acerca de este Query:

- Aunque todas las llaves primarias de la tabla SALES están especificadas en el Query el índice SALES_PK no es usado debido a que no hay una condición de límite sobre la columna principal (Company_ID) del índice SALES_PK. La única condición sobre esta columna es la condición de join.
- El optimizador podría tener seleccionada otra tabla como la tabla pivote. Si COMPANY es la tabla pivote esta podría manejar entonces un full table scan.
- En la optimización basada en reglas, si se da la oportunidad de usar el índice de la tabla pivote, la tabla pivote deberá ser la que se liste al final de la cláusula From.
- En la optimización basada en costos, el optimizador considera el tamaño de las tablas y la selectividad de sus índices mientras selecciona la tabla pivote.

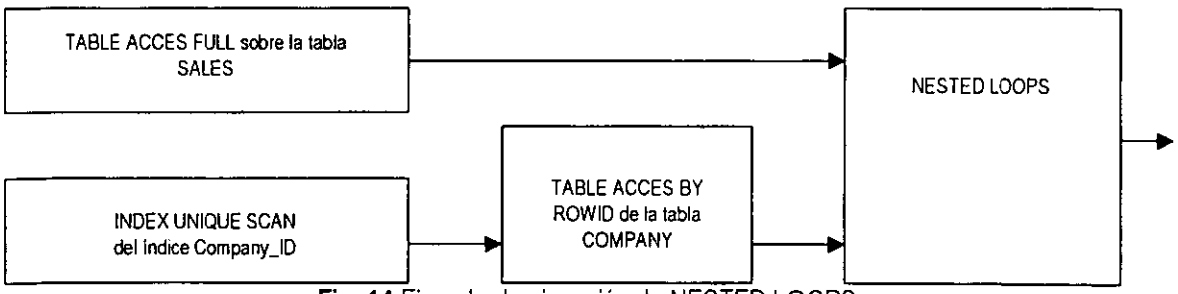


Fig. 14 Ejemplo de ejecución de NESTED LOOPS

SORT GROUP BY

SORT GROUP BY ejecuta una función de agrupamiento sobre un conjunto de registros. En el ejemplo, se muestra un Query que recupera el número de registros para cada Zip sobre la tabla COMPANY. Se recupera un registro para cada valor distinto de la columna Zip

```
Select zip, COUNT(*)
From   COMPANY
Group by Zip;
```

Plan

```
SORT GROUP BY
  TABLE ACCES FULL COMPANY
```

El plan muestra que la tabla COMPANY ha sido barrida y después de esto se hace la operación de SORT ordenando y agrupando los registros de acuerdo al valor de la columna Zip. Cada grupo será contado y el resultado será retornado al usuario.

Fig. 15 Ejemplo de SORT GROUP BY

SORT JOIN

SORT JOIN ordena un conjunto de registros que son usados por una operación de MERGE JOIN. Ejemplo

```
Select COMPANY.Name
From   COMPANY, SALES
Where  COMPANY.Company_ID = SALES.Company_ID
And    SALES.Period_ID = 3
And    SALES.Sales_Total > 1000;
```

Plan:

```
MERGE JOIN
 SORT JOIN
   TABLE ACCES FULL SALES
  SORT JOIN
   TABLE ACCES FULL COMPANY
```

La interpretación del plan coincide exactamente con la operación de MERGE JOIN, en la **figura 13**[3] se puede observar que la salida del SORT JOIN es ocupada como entrada para una operación de MERGE JOIN.

SORT ORDER BY

Esta operación es usada para ordenar el resultado de un conjunto de registros sin eliminar registros duplicados. Ejemplo:

```
Select Name
From   COMPANY
Order by Name;
```

Plan:

```
SORT ORDER BY
  TABLE ACCES FULL COMPANY
```

El plan muestra que después de que el Query es resuelto (por la operación TABLE ACCES FULL) los registros son ordenados por la operación ORDER BY antes de entregar el resultado al usuario.

[3] vid supra (Fig. 13 Ejemplo de MERGE JOIN)

Fig. 16 Ejemplo de SORT ORDER BY

SORT UNIQUE

Esta operación es usada para ordenar resultados y eliminar registros duplicados antes de una operación de MINUS, INTERSECTION o UNION. En el ejemplo se usa una operación de MINUS.

```
Select Company_ID
From   COMPANY
MINUS
Select Company_ID
From   COMPETITOR;
```

Plan:

```
PROJECTION
  MINUS
    SORT UNIQUE
      TABLE ACCES FULL COMPANY
    SORT UNIQUE
      TABLE ACCES FULL COMPETITOR
```

El plan muestra que cada uno de los querys es resuelto por separado (por medio de la operación TABLE ACCES FULL) y los registros son pasados a la operación de Sort Unique y terminar como entrada para la operación MINUS. La operación SORT UNIQUE ordena los registros y elimina los duplicados y entonces envía estos registros a la operación MINUS como se muestra el la imagen.

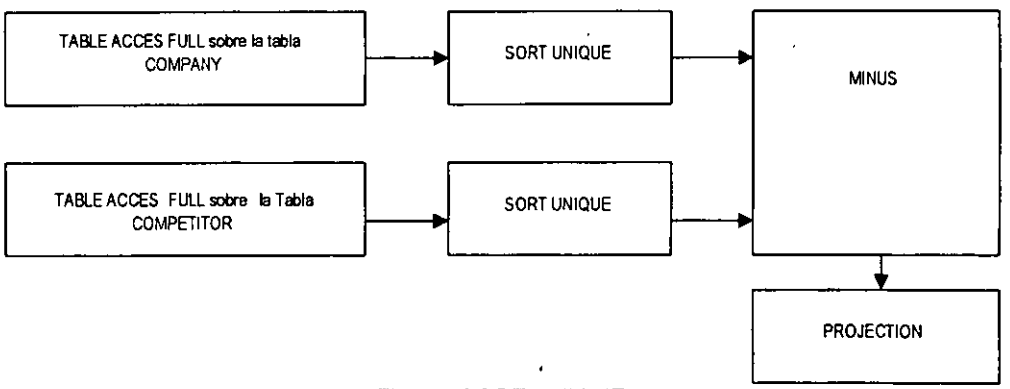


Fig. 17 SORT UNIQUE

TABLE ACCESS BY ROWID

Esta operación retorna un renglón sencillo desde una tabla basada sobre el RowID proveído por la operación. Este es un camino muy rápido para recuperar datos desde una tabla. Ejemplo;

```
Select Name
From   COMPANY
Where Company_ID = 12345
And   Activate_flag = 'Y';
```

Plan:

```
TABLE ACCESS BY ROWID COMPANY
  INDEX UNIQUE SCAN COMPANY_PK
```

Como lo muestra el plan, la columna Company_ID en la cláusula Where permite que el indice COMPANY_PK sea usado. Este índice no contiene la columna Name pero oracle puede accesar la tabla COMPANY usando el RowID retornado por el índice y así obtener el valor de Name. Un filtro implícito es ejecutado para retornar solamente los registros con Active_flag = 'Y'.

Fig. 18 TABLE ACCESS BY ROWID

TABLE ACCESS FULL

Retorna registros de una tabla cuando el RowID no está disponible. Oracle barre cada bloque en la tabla dada mientras todos los registros son leídos. Ejemplo

```
Select *
From   COMPANY;
```

Plan :

```
TABLE ACCESS FULL COMPANY
```

Todos los registros y todas las columnas son retornadas, oracle lee cada bloque y obtiene secuencialmente todos los registros.

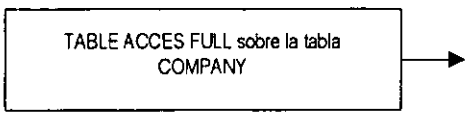


Fig. 19 Table acces full.

La mayoría de las ocasiones el plan de ejecución seleccionado por el optimizador de oracle es el mejor para resolver el Query. En un 5 ó 10 porciento de los casos sin embargo es posible que se de un mal plan de ejecución. Cuando las tablas son pequeñas el rendimiento puede disminuir generando impacto cuando estas crecen.

Como el rendimiento de un simple Query empeora rápidamente, este consume recursos que deberian estar disponibles para el resto de la aplicación, de esta manera, pronto el sistema entero tendrá un deterioro en el rendimiento. Para evitar esto, se necesita afinar el rendimiento de los querys de la aplicación.

La meta de afinar los SQLs es mejorar el rendimiento de los querys en forma manual, de esta manera el rendimiento de un Query es alterado a través de cambiar su plan de ejecución. El plan de ejecución puede ser alterado por el uso especifico de operaciones y hints[4]. Casos muy comunes donde se pueden alterar los planes de ejecución en forma manual son los siguientes:

Evitar los full table scan no planeados

Un full table scan lee secuencialmente todos los datos de una tabla, pero existen dos poderosas razones por las que no es necesario este tipo de barrido:

[4] Se puede usar hints para alterar el plan de ejecución cuando la base de datos esta configurada en modo de costos Los hints se incluyen dentro de los comandos SQL y solamente modifican el plan de ejecución para los comandos donde estos aparezcan. Los hints son colocados entre este patrón de caracteres /*+ */

- Un full table scan no es selectivo. Una búsqueda no selectiva puede ser apropiada para grandes operaciones que operen con muchos registros, por ejemplo operaciones en un segundo plano, pero esto puede ser inoperante para aplicaciones en linea.
- La lectura de datos via full table scan es removida desde los buffers del SGA rápidamente. Si la tabla comienza a ser barrida es puesta en la lista de los menos recientemente usados y por consiguiente los bloques leidos desde un full table scan son los primeros en ser removidos de el SGA. Los mismos bloques pueden ser leidos en forma múltiple fisicamente lo cual puede ocasionar cuellos de botella.

El full table scan se usa bajo las siguientes circunstancias:

- No existen indices sobre la tabla
- No existen condiciones en la cláusula Where
- No existen condiciones sobre las columnas principales indexadas
- Se usa una función sobre una columna indexada que hace que el índice se deshabilite
- Existen condiciones de diferencia (<> ó !=) sobre las columnas indexadas que nulifican el índice.

Si se usa la optimización basada en costos, oracle usará el full table scan para todos los casos anteriores, adicionalmente el optimizador basado en costos puede decidir usar un full table scan si la tabla no ha sido analizada, si la tabla es pequeña, si el índice sobre una columna no es selectivo o si se usó un hint de ALL_ROWS.

Para asegurar que un Query ocupe un índice se debe asegurar en primera instancia que todas las columnas de llave primaria y llave foránea estén indexadas. Esto hace más probable que se usen los índices durante los joins y cuando existan condiciones de limitación en el Where. De esta manera, se pueden resolver los full table scans potenciales. Oracle por default crea un índice único para cada llave primaria, pero las llaves foráneas no son indexadas automáticamente.

Usar solamente indices selectivos

La selectividad de un índice está dada de acuerdo a la cantidad de valores distintos en las columnas indexadas para el número de registros en la tabla. Si una tabla tiene 1000 registros y un índice sobre una columna de la tabla tiene 950 valores distintos entonces la selectividad del índice es de 950/1000 ó 0.95. La mejor posible selectividad es 1.00. Los indices únicos o basados en columnas no nulas siempre tienen una selectividad de 1.00.

La selectividad de un índice es una medida útil para reducir la I/O requerida por los querys. Por ejemplo, si un índice de unos 1000 registros de una tabla tiene solamente cinco distintos valores, entonces el índice es de selectividad pobre (5/1000 = 0.005) y podría hacer muy lento el Query cuando se apliquen condiciones sobre los registros. En este caso podría eficientarse el Query ejecutando una operación de full table scan en lugar de un index scan.

Manejar joins multi-tabla

Dentro de cada opción de join, existe un número de pasos que se deben dar para obtener el mejor rendimiento para un join. Si no se afinan propiamente las operaciones de NESTED LOOPS o MERGE JOIN, probablemente el rendimiento del join al principio sea aceptable pero podria incrementarse excepcionalmente cuando las tablas crezcan. El afinar los joins puede tener un impacto significativo sobre el rendimiento de los querys y existen para esto muchas opciones disponibles.

Uso de MERGE JOINS

El uso de MERGE JOINS indica que los índices no están disponibles o están deshabilitados por la sintaxis del Query. Esta operación es generalmente no apropiada para usar en aplicaciones multiusuario por las siguientes razones:

- Puede hacer muy lento el retorno del primer registro de el Query. Como esta es una operación de grupo, no retorna registros al usuario hasta que todos los registros son procesados.
- Su resultado no permanecia mucho tiempo en el SGA.
- Los segmentos temporales pueden necesitar ser reasignados para resolver el Query. Esto puede resultar en una contención entre los usuarios de los segmentos temporales.

Sin embargo, existen situaciones en las que un MERGE JOIN es el camino más eficiente para ejecutar un join. Por ejemplo, en procesos en lote o reportes grandes puede resultar la mejor opción para el Query.

Una operación de MERGE JOIN puede ser efectiva siempre y cuando un full table scan sea efectiva. Esto es en situaciones en las cuales un full table scan es preferible que el usar un indice. Un full table scan es preferible bajo dos condiciones: cuando la tabla es muy pequeña o extremadamente grande. Si la tabla es muy pequeña esta puede ser barrida más raidamente a través de un full table scan que usar un indice. Por ejemplo si la tabla esta completamente almacenada en la cantidad de bloques barridos durante una lectura a la base de datos, la tabla entera puede ser leída en una lectura fisica simple, de esta manera, un full table scan es más eficiente que un index range scan y table acces by RowID (Combinación en la que seguramente se tendría que hacer más de una lectura fisica).

Si la tabla es extremadamente grande, de la misma manera el camino más eficiente es hacer un full table scan por tres razones: La primera es dependiendo del grado en el que los datos estén almacenados físicamente y su formato y el número de registros seleccionados, se pueden hacer menos lecturas sobre los bloques para satisfacer el full table sacan que comparando con un index scan y table acces by RowID.

La segunda razón es que los bloques leídos dentro de los data block buffer del SGA por un full table scan, no están mucho tiempo en el SGA, asi, se evita que datos que no se ocupan comúnmente entre los usuarios no estén ocupando espacio.

La tercera razón es que la comparación y la porción de unión es muy eficiente. El rendimiento de los costos envueltos en un MERGE JOIN son casi enteramente encontrados en los primeros dos pasos: las operaciones de full table scan y ordenamiento. Al afinar las operaciones de MERGE JOIN deberían por consiguiente enfocarse en mejorar el rendimiento de los primeros dos pasos.

El rendimiento de un full table scan puede ser mejorado a través de afinación de I/O y mejoramiento de lecturas de bloques múltiples de oracle.

Uso de NESTED LOOPS

Una operación de NESTED LOOPS une dos fuentes de datos. Esta operación es el camino más comúnmente usado por oracle para ejecutar joins. Indica que un indice está disponible para su uso durante el join. Como es una operación de registro, los NESTED LOOPS retornan cada registro al momento de ser procesado en lugar de esperar hasta que todos sean procesados. Debido a que es una operación basada en índices, NESTED LOOPS es una operación muy efectiva de join para aplicaciones multiusuarios en línea.

Cuando se ejecuta un join NESTED LOOPS, el optimizador primero selecciona una tabla pivote y se puede dar un full table scan sobre esta tabla. Para cada registro encontrado en la tabla pivote es usado un acceso indexado a la tabla pivote para ver si este puede unir ambas tablas. Si el valor existe el registro es retornado al usuario vía operación NESTED LOOPS.

Cuando se selecciona una tabla pivote, el optimizador alinea todas las tablas en la cláusula From basándose en las condiciones del Where y los joins. El optimizador selecciona cada tabla como pivote potencial. Para cada tabla, se evalúan los posibles accesos que pueden ser usados durante el Query y puede elegir el plan de ejecución mejor de acuerdo a los índices disponibles. El optimizador basado en costos considera el tamaño de las tablas y la selectividad de sus índices. En el optimizador basado en reglas si dos o más tablas tienen rutas de acceso similares disponibles.

Para modificar un join para que se use un NESTED LOOPS, se pueden usar hints (solamente para la optimización basada en costos) o modificar manualmente la cláusula From y las condiciones del Where en el Query.

NESTED LOOPS es una operación direccional; si se unen dos tablas a través de una operación NESTED LOOPS, se obtendrán diferencias en el rendimiento dependiendo de cual de las tablas sea la pivote. Cambiando la elección de la tabla pivote se puede alterar dramáticamente el número de registros leídos por el Query y también se puede alterar dramáticamente el rendimiento del Query. Entre más y más tablas sean añadidas al Query, el pivote pasa a cada join y genera algún tipo de impacto en el rendimiento del Query.

El resultado de la primer operación NESTED LOOPS (el join entre la tabla pivote y la primera tabla manejada) es llamado el pivote principal. Si se manejan por ejemplo cuatro tablas envueltas en join de tipo NESTED LOOPS, los pasos para optimizar son:

1. Seleccionar la tabla pivote.
2. Ejecutar un join NESTED LOOPS entre la tabla pivote y la segunda tabla.
3. Ejecutar un join NESTED LOOPS entre el pivote principal retornado en el paso 2 y la tercer tabla.
4. Ejecutar un join NESTED LOOPS entre el pivote principal retornado en el paso 3 y la cuarta tabla.

Si se selecciona un mal pivote, el primer join NESTED LOOPS se ejecutará pobremente. Sin embargo, seleccionando un pivote apropiado no garantiza tampoco un buen rendimiento si se tienen tres o más tablas en el join. Si se tienen más de dos tablas en un join NESTED LOOPS es necesario considerar el tamaño del pivote principal de los registros envueltos en cada join sucesivo. Si el pivote principal no es selectivo, el rendimiento de cada join sucesivo puede ser menor, entre más tablas se agreguen el tiempo para completar el Query pude crecer excepcionalmente.

El tipo de join óptimo para una aplicación depende sobre el número de criterios (tal como el tipo de aplicación y el diseño de la base de datos). Si se usa un join NESTED LOOPS, se necesita asegurar que se de una apropiada tabla pivote (y un apropiado pivote principal). Se elige el plan de ejecución más apropiado para el Query este tendrá el mejor impacto cuando las tablas de la base de datos crezcan.

Manejo de comandos SQL que contienen vistas

Si un Query contiene una vista, entonces el optimizador tiene dos caminos para resolver el Query: el primero resuelve la vista y después resuelve el Query o bien integra el texto de la vista dentro del Query. Si la vista es resuelta primero el resultado entero de la vista es el primero en ser determinado y el resto de las condiciones del Query son aplicadas como filtro.

Dependiendo sobre el tamaño relativo de las tablas envueltas, resolver la vista primero puede causar degradaciones en el rendimiento para los querys; si la vista es integrada dentro del Query, las condiciones del Query pueden ser aplicadas dentro de la vista y el resultado más pequeño puede usarse. En algunas situaciones sin embargo se puede mejorar el rendimiento de los querys separando operaciones de grupo a través de vistas.

Si la vista contiene un conjunto de operaciones tales como GROUP BY, SUM, COUNT o DISTINCT, entonces la vista no puede ser integrada dentro del Query.

Si la vista retorna un resultado grande o el resultado de la vista se usa para ser filtrado por condiciones adicionales en el Query, el Query probablemente se beneficiaría al tener el la vista del SQL integrada en el Query. El optimizador automáticamente ejecutaría la integración si este puede. Para evitar vistas que no puedan ser integradas dentro de los querys, no se debe usar funciones de agrupamiento en estas.

Algunas veces se puede desear que el SQL de la vista no sea integrado dentro del resto del Query. Por ejemplo, si se ejecuta una operación de GROUP BY sobre un join NESTED LOOPS de dos tablas, la operación de agrupamiento no es completada mientras las dos tablas hayan sido completamente unidas.

Si se tiene una vista que no contiene una operación de agrupamiento y no se necesita que el SQL de la vista sea integrado dentro del resto del Query se tiene un hint llamado NO_MERGE que puede ser de ayuda para prevenir que el Query de la vista se una al resto del Query.

Afinar subquerys

Cuando se usan subquerys se pueden encontrar diferentes problemas como los siguientes:

- Los subquerys pueden ser resueltos antes que el resto de el Query
- Los subquerys pueden requerir hints especificos que no están directamente relacionados con le Query que llama al subquery

- Los subquerys que pueden ser ejecutados como un simple Query pueden en cambio ser escritos como distintos subquerys
- Los subquerys no pueden ejecutar chequeos de existencia en forma más eficiente, o bien usan una cláusula NOT IN o fallan al usar una cláusula EXISTS

Si un Query contiene un subquery, el optimizador tiene dos caminos para resolver el Query: primero se resuelve el subquery y después se resuelve el Query, o bien se integra el subquery dentro del Query. Si el subquery es resuelto primero el resultado entero del subquery es primeramente calculado y el resto de las condiciones del Query son aplicadas como filtro. Si el subquery es integrado dentro del Query, las condiciones del subquery y las tablas pueden ser unidas al resto del Query. Si no se usan subquerys para ejecutar chequeos de existencia, el método del join se podrá ejecutar mejor que el de la vista.

Si el subquery contiene un conjunto de operaciones tales como GROUP BY, SUM o DISTINCT, entonces el subquery no puede ser integrado al resto del Query.

Comúnmente los subquerys no retornan registros, pero ejecutan una validación de datos. Por ejemplo, se puede checar que una tabla con una llave primaria no tenga registros hijos en una llave foránea de tabla para valores específicos sobre la columna de llave primaria. Los registros relacionados a tablas o existen o no existen, estos son los llamados chequeos de existencia. Se pueden usar los operadores de Exists o Not Exists para mejorar el rendimiento de chequeos de existencia.

Se debe usar EXISTS siempre y cuando un join no retorne alguna columna de una de las tablas, si solamente se desea saber si el registro existe, de esta manera eliminamos accesos innecesarios a tablas. Se debe usar Not Exists para sustituir una cláusula NOT IN que precede a un subquey.

Manejo de acceso a tablas muy grandes

Como una tabla crece en forma significativa, puede entonces llegar a ser más grande que el tamaño de los data block buffer cache en el SGA , debido a esto, se necesita afinar los querys obtenidos de esa tabla con perspectiva diferente. Considerando que usuarios múltiples puedan beneficiarse al compartir datos de tablas pequeñas en el SGA, estos beneficios desaparecen cuando tablas muy grandes son accesadas.

Cuando una tabla y sus índices son pequeños, estos pueden ser compartidos comúnmente en el SGA, distintos usuarios ejecutan lecturas de la tabla o los index range scans pueden usar los mismos bloques una y otra vez; como resultado de rehusar los bloques dentro del SGA, la medida de rehuso dentro del SGA se incrementa, la tabla crece y sus índices crecen también y como estos toman proporciones mayores al espacio disponible en el SGA, se hace poco probable que el siguiente registro se encuentre dentro del SGA, resultado en que para cada lectura lógica se requiera también de una lectura física.

El SGA está diseñado para un máximo de rehuso de los bloques leídos desde los datafiles. Para hacer esto, el SGA mantiene una lista de los bloques que han sido leídos; si los bloques fueron leídos a través de un índice o a través de un acceso por RowID, estos bloques son guardados en el SGA por un gran tiempo. Si un bloque fue leído a través de un full table scan, este bloque es el primero en ser removido del SGA cuando se necesita más espacio en el data block buffer cache.

Para aplicaciones con tablas pequeñas, el manejo del data block buffer caché en el SGA maximiza su rehuso de bloques. Si un index range scan es ejecutado sobre una tabla muy grande, los bloques del índice son guardados por un gran tiempo en el SGA, aunque es probable que otros usuarios no puedan usar los valores de estos bloques de índices. Si un índice es grande muchos de sus bloques pueden ser leídos y a su vez consumir una importante porción del espacio disponible en los datablocks buffer cahe del SGA. Una mayor cantidad de espacio puede ser consumida por los bloques que accesan a las tablas a través del RowID y estos bloques tienen menor probabilidad de ser rehusados.

Si se va a usar un index scan para una tabla grande, no se puede asumir que el index scan pueda ejecutarse mejor que un full table scan. Un unique scan o range scan de índices que no son seguidos por un acceso a tablas se ejecutan bien, pero un range scan de un índice seguido por un acceso a tabla por RowID puede ejecutarse pobremente. Como la tabla crece significativamente y adquiere un tamaño mayor al data block buffer cache, el punto de rompimiento entre el index scan y full table scan decrece. Si se lee más de uno por ciento de registros entre unos 10,000,000 registros de una tabla, es mejor un full table scan que un index range scan y un table acces by RowID.

Para mejorar el uso de indices durante grandes accesos, se necesita eliminar dos operaciones: range scans y accesos subsecuentes a tablas. El costo de acceso solo a índices para el SGA puede ser alto. El indexar una tabla completamente es útil solo si los datos de la tabla son bastante estáticos. Durante un Query, todos los datos requeridos por este pueden ser proveídos a través de un índice y el acceso a tablas puede ser no necesario.

En un hash cluster, el orden en el cual los registros son insertados en la tabla no importa, la ubicación fisica de un registro está determinada por sus valores en columnas llave. Una hashing function es aplicada a los valores llave para el registro y oracle usa el resultado para determinar en cual bloque se almacenará el registro. Los registros dentro del hash cluster son rápidamente cubiertos por todos los bloques asignados. El uso de has clusters elimina la necesidad de un índice para un valor simple de llave primaria.

6. AFINAR EL CPU Y ARQUITECTURA DEL SISTEMA

Al establecer un examen apropiado de la cantidad de CPU que el sistema debería utilizar se puede entonces distinguir cuando la cantidad no es suficiente, o si un sistema consume mucho CPU. Se puede empezar esta tarea por determinar tres situaciones: la cantidad de CPU que oracle utiliza mientras la máquina está inactiva, cuando la carga de trabajo es media y cuando la carga de trabajo está en horas pico

La carga de trabajo es un factor importante cuando se evalúa el nivel de utilización de CPU por el sistema. Durante las horas pico, se usa un 90% de CPU con un 10% inactivo y en espera de que pueda ser aceptable. Un treinta por ciento de utilización en tiempos de carga baja puede ser entendible. Sin embargo si un sistema muestra una alta utilización en horas normales entonces no será aceptable en horas pico.

Por ejemplo, en la **figura 20** podemos observar una ilustración que tiene periodos pico entre las 10:00 am y las 2:00 pm

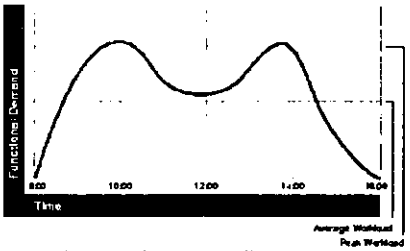


Fig. 20 Carga de trabajo promedio y carga de horas pico

En este ejemplo la aplicación tiene 100 usuarios trabajando 8 horas al día. Si un usuario genera una transacción cada 5 minutos esto podría significar 9,600 transacciones por día. En un curso de 8 horas el sistema soporta 1200 transacciones por hora lo cual significa un promedio de 20 transacciones por minuto. Si esta demanda permanece constante se puede construir un sistema que mantenga este promedio de carga de trabajo. Sin embargo, en la gráfica podemos observar que en realidad se necesitan 120 transacciones por minuto, entonces se debe configurar un sistema que pueda soportar esta carga de trabajo para horas pico.

Las estadisticas de oracle reportan solamente la utilización de CPU de las sesiones de oracle y se debe considerar que cada proceso que corre en el sistema afecta la cantidad de recursos de CPU disponible. Afinando factores no-oracle puede traer como resultado un mejor rendimiento en oracle

Hay que asegurarse que la memoria es suficiente para las cargas de trabajo , de manera que la máquina no genere una carga muy pesada (es decir que se generen procesos de alternancia y paginación fuera de memoria). El sistema operativo asigna espacios fijos de tiempo durante los cuales los recursos del CPU están disponibles para los procesos. Si los procesos malgastan cantidades grandes de tiempo en verificar si hay espacio suficiente para correr procesos, entonces se requerirán nuevos componentes en esa máquina ya que solamente usa el 50% del tiempo repartido para actualizar el trabajo actual.

La lentitud de un sistema puede resultar en una carga excesiva para el CPU, comúnmente una aplicación puede generar un mensaje que necesita ser enviado a través de la red una y otra vez. Esto provoca que se tenga que generar mucho trabajo cada que este es enviado. Para dar solución a este problema se pueden dejar una cantidad significativa de esto mensajes en lote y enviar uno solo más grande y hacer solo una vez el trabajo, reduciendo así el uso de recursos

Utilización de CPU por oracle

La vista V$SYSSTAT muestra el uso de CPU por oracle en todas las sesiones. Las estadísticas de "CPU usado por esta sesión" muestra el CPU agregado usado por todas las sesiones. Se puede entonces usar esta vista para ver que sesiones en particular usan más CPU. Por ejemplo, si se tienen 8 CPUs, entonces para algún minuto dado en tiempo real se tienen 8 minutos de tiempo de CPU disponible. En sistemas NT y UNIX este puede ser tiempo de usuario o bien tiempo en modo de sistema. Si los procesos no están corriendo entonces están en espera. El CPU utilizado por todos los sistemas puede ser de más de un minuto por intervalo.

En algún momento dado se debe conocer que tanto tiempo oracle ha utilizado el sistema. Así, si se tienen 8 minutos de tiempo disponible y oracle usa 4 minutos, entonces se sabrá que el 50% de todo el tiempo de CPU es usado por oracle. Si los procesos no consumen ese tiempo entonces algunos otros procesos lo hacen. Hay que regresar al sistema y verificar entre los procesos que usan el CPU identificar este y determinar el motivo por el que usa mucho CPU, de esta manera se puede afinar ese proceso.

Las áreas que se necesitan checar en oracle por la utilización de CPU son:
- Reparseo de comandos SQL
- Comandos SQL ineficientes
- Consistencia de lectura
- Limitaciones de escalabilidad dentro de una aplicación

Reparseo de comandos SQL

Cuando se tienen compartidos comandos SQL que son ineficientes puede resultar esto en un reparseo. Podemos checar la vista V$SYSSTAT para verificar si el parseo en general es el problema:

```
SELECT * FROM V$SYSSTAT
WHERE NAME IN ('parse time cpu', 'parse time elapsed', 'parse count');
```

Para interpretar las estadisticas hay que tener en cuenta lo siguiente:

- response time = service time + wait time
- response time = elapsed time
- service time = CPU time, therefore
- elapsed time - CPU time = wait time

Así se puede detectar el tiempo de respuesta sobre el parseo. Después se puede hacer un Query sobre la vista V$SQLAREA para encontrar todos los comandos individuales que son reparseados frecuentemente

```
SELECT SQL_TEXT, PARSE_CALLS, EXECUTIONS FROM V$SQLAREA
ORDER BY PARSE_CALLS;
```

Ya que se ha identificado el problema en los comandos, se tienen las siguientes opciones para afinar esto.

- Reescribir la aplicación de tal suerte que los comandos no se reparseen continuamente
- Si esto no es posible, se debe reducir el parseo usando el parámetro de inicialización SESSION_CACHED_CURSORS
- Si la cuenta del parseo es pequeña y la cuenta de ejecución también lo es y además varios de los comandos SQL son muy similares excepto en las cláusulas Where se pueden generar códigos que usen variables para englobarlos y así reducir la cantidad de parseo

Comandos SQL ineficientes

Comandos SQL ineficientes pueden consumir grandes cantidades de CPU, para detectar estos, se puede consultar el siguiente comando:

```
SELECT BUFFER_GETS, EXECUTIONS, SQL_TEXT FROM V$SQLAREA;
```

Los comandos SQL con un número grande de buffers pueden ser ineficientes. Se puede reducir el uso de CPU afinando estos comandos.

Consistencia de lectura

Un sistema podría gastar un gran tiempo en hacer Rollback o deshaciendo cambios a bloques para generar una vista consistente

- Si existen muchas transacciones pequeñas y estas están activas en segundo plano en las mismas tablas donde se inserta comúnmente, los querys pueden tener muchos cambios para hacer Rollback .
- Si el número de segmentos de Rollback es pequeño, se puede gastar mucho tiempo generando un Rollback de todas esas transacciones
- Los querys han comenzado hace un gran tiempo debido a que la cantidad de segmentos de Rollback es muy pequeño para las transacciones. Sería mejor generar más segmentos de Rollback o incrementar el rango del commit, es decir, si corren juntos diez proceso, se usa solamente un commit entonces se reducirá el número de transacciones en un factor de diez
- Se puede también hacer un barrido de muchos de los buffers en primer plano y encontrar un buffer libre. Este gasta recursos de CPU; para arreglar este problema hay que afinar el proceso DBWR para que escriba más frecuentemente.

Limitaciones de escalabilidad dentro de una aplicación

En la mayoría de las discusiones sobre afinación de CPU se asume que existe una escalabilidad lineal pero este realmente nunca es el caso. Se pueden generar problemas en la aplicación que pueden dañar la escalabilidad, por ejemplo la inclusión de muchos índices que tengan problemas, muchos datos en los bloques, datos no particionados. Los problemas de contención generan pérdidas de ciclos de CPU y no permiten que la aplicación tenga una escalabilidad lineal.

Problemas de CPU y arquitectura del sistema

Si ha hecho más grande el límite de CPU para tener más poder disponible en el sistema, si también se han agotado todos los recursos de afinación del CPU, entonces se debería considerar una reestructuración de la arquitectura del sistema. Se debe considerar si se cambia a otra arquitectura diferente para dar por resultado un CPU más adecuado

Si se corre un sistema multitarea hay que checar los niveles de uso de CPU. Por ejemplo, en un sistema de tres clientes que el servidor generalmente está inactivo y el cliente está completamente ocupado. En esta situación el servidor no tiene problemas de rendimiento, generalmente es el cliente quien los tiene. Se debe considerar hacer cambios en el cliente a fin de que se resuelvan problemas de CPU.

7. AFINAR LAS ASIGNACIONES DE MEMORIA

Oracle almacena información tanto en la memoria como en el disco. El acceso a memoria es mucho más rápido que el disco, esta es la mejor área para manejar los requerimientos de los datos, por eso, para mejorar el rendimiento se deben almacenar tantos datos como sea posible en la memoria. Sin embargo los recursos de la memoria en un sistema operativo está limitado, debido a esto, hay que afinar las asignaciones de memoria disponibles como estructuras de memoria de oracle.

Debido a que los requerimientos de memoria dependen de las aplicaciones, se deben afinar las asignaciones de memoria después de afinar la aplicación y los comandos SQL. Afinar las asignaciones de memora antes de afinar la aplicación y los SQLs puede generar que crezcan en tamaño algunas estructuras de memoria de oracle que no necesitaban necesariamente ser modificadas.

Cuando se usan herramientas de un sistema operativo tales como ps –efl o ps –aux de UNIX se puede observar el tamaño de los procesos de oracle, se puede notar así que los procesos son muy grandes. Para interpretar las estadísticas mostradas se debe determinar que tanta cantidad de esos procesos esta atribuida a memoria compartida y que cantidad de la memoria dada consume el proceso. De aquí podemos derivar diferentes afinaciones para áreas específicas de memoria asignada a las estructuras de oracle.

- Afinar los requerimientos de memoria del sistema operativo
- Afinar el Redo Log Buffer
- Afinar las Private SQL y PL/SQL Areas
- Afinar la Shared Pool
- Afinar el Buffer Cache
- Afinar las Sort Areas
- Reasignación de memoria
- Reducir el total de memoria usada

Afinar los requerimientos de memoria del sistema operativo

El propósito del SGA es almacenar datos en memoria para un rápido acceso, el SGA deberia siempre estar contenido en la memoria principal. Si partes del SGA son puestas en el disco estos datos no serán accesibles rápidamente, en muchos sistemas operativos la desventaja de paginación excesiva es de mayor peso que una ventaja de SGA grande.

Aunque es mejor guardar el SGA en memoria, el contenido del SGA estará dividido en áreas calientes y partes frías. Las áreas calientes siempre estarán en memoria debido a que siempre se hace referencia a ellas. Las partes frías pueden ser puestas y pueden generar una penalidad en su rendimiento si estas se regresan a las áreas calientes, tenemos aquí un problema inminente de rendimiento si las partes calientes no pueden quedar en memoria.

Algunos sistemas operativos como las computadoras IBM mainfame están equipados con almacenamiento expandible o memoria espacial adicional a la memoria principal para la cual la paginación puede ser ejecutada realmente rápido. Estos sistemas operativos deben ser capaces de paginar los datos entre la memoria principal y la memoria expandida tan rápido que oracle pueda leer y escribir datos entre el SGA y el disco. Por esta razón se puede permitir un SGA grande que pueda ser cambiado y asegurar un mejor rendimiento del SGA pequeño que queda en la memoria principal. Si el sistema operativo tiene memoria extendida, se puede aprovechar para asignar un SGA más grande a pesar de la paginación resultante.

Asignación de memoria a usuarios individuales.

En algunos sistemas operativos se puede tener el control sobre la cantidad de memoria física asignada a cada usuario. Se debe asegurar que todos los usuarios tengan asignada suficiente memoria para los recursos que ellos necesitan al usar sus aplicaciones con oracle. Dependiendo del sistema operativo los recursos pueden incluir el SGA, herramientas de aplicación y datos de aplicaciones específicas, y así se puede aumentar o reducir la cantidad de memoria requerida por cada usuario.

Afinar el Redo Log Buffer

El parámetro LOG_BUFFER reserva espacio para el redo log buffer el cual es de tamaño fijo. En máquinas con procesadores rápidos y discos relativamente lentos los procesos pueden llenar el resto de los buffer en el tiempo en que el redo log writer mueve una porción del buffer al disco. El log writer siempre es iniciado cuando el buffer comienza a llenarse. Por esta razón un buffer grande hace menos entradas posibles por que las nuevas entradas chocan con la parte del buffer que aun se está escribiendo.

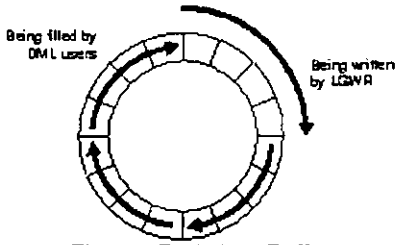


Fig. 21 Redo Log Buffer

El log buffer generalmente es pequeño en comparación con el tamaño del SGA y un incremento modesto puede ser muy significativo. La proporción del log buffer es el requerimiento de espacio para redo log entre las entradas redo. Si la proporción es más grande de 1:5000 entonces se debe incrementar el tamaño del redo log buffer hasta que se satisfaga la demanda.

Afinar Private SQL y PL/SQL Areas

Hay una relación estrecha entre la memoria y el reparseo. Si existe mucho reparseo, se requiere menos memoria. Si se reduce el reparseo (por crear más comandos SQL), entonces el requerimiento de memoria en el cliente se incrementa. Esto se debe principalmente al número de cursores abiertos.

Para afinar private SQL áreas se requiere identificar parseos innecesarios llamados desde alguna aplicación y reducir estos. Para reducir el parseo se tiene que incrementar el tamaño de las private SQL áreas que la aplicación puede tener asignadas.

Identificar llamadas innecesarias de parseo

Técnica 1

Un camino para identificar llamadas de parseo innecesarias es correr una aplicación con el SQL trace habilitada. Para cada comando SQL en la salida del trace, se debe examinar la estadística count. Esta columna dice en que tanto tiempo la aplicación hace una llamada de parseo para el comando. La estadística incluye llamadas de parseo (parse calls) que se satisfacen por acceso al library cache. Si el valor count para la fase de parseo está cerca del valor de count para la fase de ejecución de un comando, la aplicación puede deliberadamente hacer una llamada de parseo cada momento que se ejecute este comando, se debe para este caso intentar reducir las llamadas de parseo a través de las herramientas de la aplicación.

Técnica 2

Otro camino para identificar las llamadas de parseo es checar la vista V$SQLAREA usando el siguiente Query

```
SELECT sql_text, parse_count, executions
FROM V$SQLAREA
```

Cuando el valor parse_count, esta cerca de valor executions, para un Query dado, puede ser que continuamente se este parseado un comando SQL en particular

Técnica 3

Se puede también identificar llamadas innecesarias de parseo identificando la sesión que las dispara ejecutando el siguiente Query:

```
SELECT * FROM V$STATNAME
WHERE name in ('parse_count','execute_count')
```

Los resultados del Query serían algo parecido a esto:

```
statistic#,  name
-----------  ---------
100          parse_count
90           execute_count
```

Entonces correr el siguiente Query:

```
SELECT * FROM V$SESSTAT
WHERE statistics# in (90,100)
ORDER BY value, sid;
```

El resultado será una lista de todas las sesiones y la cantidad de reparseo de ellas. Para cada identificador de sistema (sid), hay que consultar la tabla de V$SESSION para encontrar el nombre del programa que causa el problema

Reducir llamadas innecesarias de parseo

Dependiendo de que aplicación de oracle se esté utilizando, se debe ser capaz de controlar la frecuencia de las llamadas de parseo y asignar y desasignar private SQL áreas. Si la aplicación rehusa las private SQL áreas para múltiples comandos SQL se determina que tantas llamadas de parseo se ejecutan en la aplicación y que tantas private SQL áreas requiere la aplicación

En general, una aplicación que rehusa private SQL áreas para múltiples comandos SQL no necesita de muchas private SQL áreas como una aplicación que no las rehusa, sin embargo una aplicación que rehusa private SQL áreas debe ejecutar más llamadas de parseo debido a que la aplicación debe hacer un nuevo parseo siempre que exista un private SQL es rehusado por un nuevo comando SQL

Para asegurar que una aplicación pueda abrir suficientes SQL áreas para acomodar todos sus comandos SQL, se necesita asignar más private SQL areas y puede además necesitar incrementarse el límite de cursores permitidos por sesión.

Afinar la shared pool

Debido al algoritmo que oracle usa para manejar datos en la shared pool se cerciora que se mantendrá el diccionario de datos en memoria más grande que los datos del library cache, afinar el library cache asegura que los datos del dictionary caché son aceptables. Asignar espacio en el shared pool para información de sesión es solamente necesario si se usa un servidor con arquitectura cliente-servidor.

En la shared pool, algunos de los caches son dinámicos y estos crecen o decrecen como sea necesario. Estos caches dinámicos incluyen el library cache y el data dictionary cache. Los objetos pueden ser puestos fuera de estos caches si no hay mas espacio en el shared pool. Por esta razón se debe incrementar el tamaño de la shared pool si el conjunto de datos no encaja dentro de esta. El cache miss sobre el data dictionary cache o el library cache es más caro que un miss sobre el buffer cache. Por esta razón se debe asignar suficiente memoria para el shared pool.

Para muchas aplicaciones el tamaño de la shared pool es critico para el rendimiento de oracle. La shared pool mantiene tanto el diccionario de datos y las representaciones completas de bloques de PL/SQL compilados y comandos SQL. Los bloques de PL/SQL incluyen procedimientos, funciones, paquetes, triggers y bloques anónimos de PL/SQL. El tamaño de la shared pool es menos importante solo para aplicaciones que usan un muy limitado número de comandos SQL

Si la shared pool es pequeña, entonces el servidor debe dedicar recursos para el manejo limitado de espacio disponible. Este consume CPU y causa contención. Entre más se usan en triggers y procedimientos almacenados, más grande debe ser la shared pool. Esta seria de algunas decenas de megabytes

Siempre es mejor medir las estadísticas sobre un periodo específico de tiempo, se puede determinar la proporción library cache y del row cache con los siguientes querys:

```
Select (sum(pins-reloads))/sum(pins) "Lib Cache"
from v$librarycache;

select (sum(gets-getmisses-usage- fixed))/sum(gets) "Row Cache"
from v$rowcache;
```

El resultado muestra la dimensión del library cache (En forma general el número de reparseos reflejados en el library cache). La cantidad de memoria libre en la shared pool es reportada en la vista V$SGASTAT. Los valores instantáneos pueden ser reportados usando el siguiente Query

```
select * from v$sgastat where name = `free memory';
```

Si esta siempre tiene memoria libre disponible dentro de la shared pool, entonces incrementando el tamaño del pool se tendrían beneficios pequeños o definitivamente no se tendría ningún efecto benéfico.

Una vez que una entrada ha sido movida al shared pool esta no puede ser movida. Esto puede causar que el pool se pueda fragmentar. En un servidor UNIX, se puede usar el paquete PL/SQL DBMS_SHARED_POOL para manejar la shared pool. PL/SQL tiende a asignar grandes objetos, si se ordenan grandes grupos, estos son movidos fuera de la shared pool cuando pequeños grupos son puestos allí. Usando el paquete DBMS_SHARED_POOL se pueden guardar permanentemente los objetos grandes en el shared pool.

Si la memoria libre está cerca de cero y la dimensión de library cache o el row cahe es menor que 0.95 entonces se debe incrementar el tamaño de la shared pool hasta que mejore.

Afinar el Library Cache

El library cache contiene las shared SQL y PL/SQL áreas. En un comando SQL que esta contenido en el library cache misses se generan dos pasos para poder procesarlo:

Parseo. Si una aplicación hace una llamada de parseo para un comando SQL y la representación del parseo para ese comando aun no existe en la shared SQL área en el library cache, oracle parsea el comando y lo coloca en la shared SQL área.

Ejecución. Si una aplicación hace una llamada de ejecución para un comando SQL en el shared SQL área que ya no esta contenido en el library cache, oracle implícitamente reparsea el comando y lo asigna en una nueva shared SQL área y lo ejecuta.

Se pueden monitorear las estadísticas reflejando la actividad del library cache examinando la tabla V$LIBRARYCACHE. Estas estadísticas reflejan toda la actividad del library cache de la instancia más recientemente iniciada. Por default esta tabla está solamente disponible para el usuario SYS y para los usuarios que tengan el privilegio de sistema SELECT ANY TABLE como SYSTEM. Cada registro de esta tabla contienen estadísticas para un tipo de ítem guardado en el library cahe. El ítem descrito por cada registro es identificado por el valor de la columna NAMESPACE. Los registros de la tabla reflejan la actividad del los comandos SQL y los bloques de PL/SQL.

- SQL AREA
- TABLE/PROCEDURE
- BODY
- TRIGGER

Los registros con otros valores para NAMESPACE reflejados por la actividad del library cache son definiciones que oracle usa para su mantenimiento.

Estas columnas en la tabla V$LIBRARYCACHE reflejan el library cahe misses sobre las llamadas de ejecución

PINS Esta columna muestra la cantidad de tiempo en que se ejecuta un ítem en el library cache
RELOADS Esta columna muestra la cantidad de library cache misses sobre pasos de ejecución

Si se hace un Query sobre la tabla V$LIBRARYCACHE se pueden monitorear las estadísticas sobre un periodo de tiempo

```
SELECT SUM(pins) "Executions",
SUM(reloads) "Cache Misses while Executing"
FROM v$librarycache;
```

Esto es lo que genera como salida el Query

```
Executions Cache Misses while Executing
---------- ----------------------------
  320871                    549
```

Examinando los datos retornados por el Query se pueden hacer las siguientes observaciones:

- Las suma de la columna PINS indica que los comandos SQL y los bloques de PL/SQL y la definición de los objetos fueron accesados en un tiempo total de 320,871
- La suma de la columna RELOADS indica que 549 de esas ejecuciones causan que oracle implicitamente reparsee un comando o bloque o cargue nuevamente una definición de objeto por que ya estaba fuera del library cache
- La dimensión del total de RELOADS para el total de PINS es cerca del 0.17% que significa que solamente esa cantidad de ejecuciones resulto en reparseo.

El total de RELOADS debería ser cercano a cero, si el tamaño de RELOADS para PINS es mayor que 1%, entonces se debe reducir el library cache misses. Se puede reducir el library cache misses sobre las llamadas de ejecución asignando más memoria para el library cache. Para asegurar que las shared SQL áreas queden en el cache una vez que los comandos has sido pareseados, se debe incrementar la cantidad de memoria disponible para el library cahe hasta que el valor de V$LIBRARYCACHE.RELOADS este cerca de cero. Para incrementar la cantidad de memoria disponible para el library cache, se debe incrementar el valor del parámetro de inicialización SHARED_POOL_SIZE. El valor máximo para este parámetro depende del sistema operativo. Esta medida podría reducir el reparseo implicito de los comandos SQL y los bloques de PL/SQL en ejecución.

Para tener una ventaja mayor para la memoria adicional disponible para las shared SQL áreas, se necesitaria también incrementar la cantidad de cursores permitidos por sesión. Se puede incrementar este limite por medio del parámetro de inicialización OPEN_CURSORS.

Otra forma de reducir el library cache misses sobre las llamadas de parseo es asegurando que comandos SQL y bloques de PL/SQL usen la shared SQL área siempre. Es decir, para dos diferentes ocurrencias de un comando SQL o un bloque de PL/SQL se use una misma shared SQL área, se deben acordar criterios idénticos. Por ejemplo:

- El texto de los comandos SQL o bloques de PL/SQL deben ser idénticos, caracter por caracter, incluyendo espacios y mayúsculas/minúsculas. Por ejemplo, los siguientes comandos SQL no usan la misma shared SQL área

```
SELECT * FROM emp;
SELECT *   FROM emp;
```

- Referencias a objetos del esquema en comandos SQL o bloques de PL/SQL deben ser resueltos en el mismo objeto en el mismo esquema. Por ejemplo, si los esquemas de los usuarios BOB y ED contienen la tabla EMP y ambos usan el siguiente comando SQL, estos comandos no pueden usar la misma shared SQL área.

```
SELECT * FROM emp;
SELECT * FROM emp;
```

Si ambos querys usan la misma tabla y agregan a la tabla ese esquema, como en el siguiente comando, entonces estos pueden usar la misma shared SQL área:

```
SELECT * FROM bob.emp;
```

- Las variables en los comandos SQL deben emparejar el nombre y el tipo de datos. Por ejemplo, estos comandos no usan la misma shared SQL área:

```
SELECT * FROM emp WHERE deptno = :department_no;
SELECT * FROM emp WHERE deptno = :d_no;
```

- Los comandos SQL y los bloques de PL/SQL deben estar optimizados.

Las shared SQL áreas son más útiles al reducir el library cache misses para múltiples usuarios corriendo la misma aplicación. Los desarrolladores deben generar estrategias para que sus aplicaciones usen las mismas shared SQL áreas, algunas de estas estrategias podrían ser las siguientes:

- Uso de constantes en las declaraciones lo menos posible.

Por ejemplo los siguiente comandos no pueden usar la misma shared SQL área debido a que no son idénticos carácter por carácter

```
SELECT ename, empno FROM emp WHERE deptno = 10;
SELECT ename, empno FROM emp WHERE deptno = 20;
```

Se puede solucionar esta cuestión su se usa una variable que tenga un valor de 10 para una ocurrencia y 20 para una segunda ocurrencia.

```
SELECT ename, empno FROM emp WHERE deptno = :department_no;
```

Las dos ocurrencias para el comando pueden ser usadas por la misma shared SQL área

- Asegurar que los usuarios de la aplicación no cambien la optimización por sesiones individuales

Se puede también incrementar la posibilidad de que los comandos usados por diferentes aplicaciones puedan compartir SQL áreas estableciendo políticas entre los desarrolladores de las aplicaciones, por ejemplo:

- Estandarizar las convenciones de nombramientos de variables y espacios para comandos SQL y bloques de PL/SQL
- Usar procedimientos almacenados siempre que sea posible. Múltiples usuarios usan los mismos procedimientos almacenados y automáticamente usan la misma shared SQL área.

Si no se tiene una library cache misses, se puede tener velocidad en las llamadas de ejecución modificando el valor del parámetro de inicialización CURSOR_SPACE_FOR_TIME. Este parámetro especifica cuando una shared SQL área puede ser desasignada de el library cache para hacer espacio para un nuevo comando SQL. El valor de default para este parámetro es FALSE, esto significa que la shared SQL área puede ser desasignada desde el library cache indiferentemente cuando otro cursor es abierto. El valor TRUE significa que la shared SQL área puede solamente ser desasignada cuando los cursores de la aplicación se cierren.

Dependiendo del valor de CURSOR_SPACE_FOR_TIME, oracle se comporta de diferente manera cuando una aplicación tiene una llamada de ejecución. Si el valor es FALSE, oracle debe tener tiempo para checar que la shared SQL área contenía el comando SQL en el library cache. Si el valor es TRUE, oracle no hace este chequeo por que la shared SQL área nunca puede ser desasignada mientras un cursor asociado a la aplicación esta abierto.

No se fija el valor del CURSOR_SPACE_FOR_TIME a TRUE si hay library cache misses en llamadas de ejecución. Las library cache misses indican que el shared pool no es lo suficientemente grande para mantener las shared SQL áreas de todos los cursores actualmente abiertos. Si el valor es TRUE y no hay espacio suficiente en la shared pool para un nuevo comando SQL, el comando no puede ser parseado y oracle retorna un error diciendo que no hay más memoria disponible. Si el valor es FALSE y no hay espacio para un nuevo comando, oracle desasigna una shared SQL área existente.

No se fija el valor de CURSOR_SPACE_FOR_TIME a TRUE si la cantidad de memoria disponible es escasa para cada usuario de private SQL áreas. El valor también previene la desasignación de private SQL áreas asociadas con cursores abiertos. Si las private SQL áreas para todos los cursores abiertos actualmente se llenan de tal suerte que no hay memoria suficiente para asignar una

private SQL área para un nuevo comando SQL, el comando no puede ser parseado y oracle retorna un error indicando que no hay memoria suficiente

Cursores de Sesión

Si una aplicación continuamente usa llamadas de parseo sobre el mismo comando SQL, el reabrir los cursores de sesión puede afectar el rendimiento del sistema. Los cursores de sesión pueden ser almacenados en un session cursor cache. Esta característica puede ser particularmente usada para aplicaciones diseñadas usando oracle Forms debido al swicheo entre todos los cursores de sesión asociados con la forma.

Oracle usa la shared SQL área para determinar si más de tres parseos requeridos han sido usados por un comando dado y también oracle asume que los cursores asociados a la sesión deben ser movidos al session cursor cache. Los subsecuentes requerimientos de parseo por la misma sesión se encontrarán en el session cursor cache.

Para habilitar esta característica se debe dar un valor al parámetro de inicialización SESSION_CACHED_CURSORS. Este parámetro es un entero positivo que especifica el número máximo de cursores de sesión guardados en el cache.

Para determinar cuando el sesión cursor cache es suficientemente grande para la instancia, se pueden examinar las estadísticas de sesión en la vista V$SESSTAT. Estas estadísticas cuentan la cantidad de tiempo de llamadas de pareso de un cursor en el sesión cursor cache. Si las estadísticas muestran un porcentaje relativamente bajo del total, se debería considerar fijar un valor grande para el parámetro SESSION_CACHED_CURSORS.

Afinación del Data Dictionary Cache

Para determinar si los misses del data dictionary cache están afectando el rendimiento de oracle, se puede examinar la actividad del cache haciendo una consulta sobre la tabla V$ROWCACHE. Los misses sobre el data dictionary cache pueden ser esperados en algunos casos, cuando la instancia se inicia el diccionario de datos puede no contener datos y algunos comandos SQL usados es probable que resulten en cache misses. Conforme más datos sean leídos en el cache, la probabilidad de los cache misses puede decrecer.

La vista V$ROWCACHE muestra estadísticas que reflejan actividad del diccionario de datos. Por default esta tabla está solamente disponible para el usuario SYS y para los usuarios con el privilegio de sistema SELECT ANY TABLE. El siguiente Query monitorea las estadísticas en la tabla V$ROWCACHE para un periodo de tiempo mientras una aplicación está corriendo:

```
SELECT SUM(gets)  "Data Dictionary Gets"
      ,SUM(getmisses) "Data Dictionary Cache Get Misses"
FROM v$rowcache;
```

Este Query genera la siguiente salida:

```
Data Dictionary Gets  Data Dictionary Cache Get Misses
--------------------  --------------------------------
             1439044                              3120
```

Examinando los datos retornados por el Query se pueden hacer las siguientes observaciones:

- La suma de la columna GETS indica que se tienen un total de 1,439,044 requerimientos para el diccionario de datos
- La suma de la columna GETMISSES indica que 3120 de estos requerimientos resultaron en cache misses
- La relación entre las sumas de GETMISSES para GETS es cerca del 0.2%

Para accesos frecuentes al dictionary cache, la relación total entre GETMISSES y el total de GETS debería ser menor que 10% o 15%. Si el promedio continua incrementándose mientras la aplicación está corriendo, se debería considerar incrementar la cantidad

de memoria disponible para el data dictionary cache. Para incrementar la memoria disponible para el cache, se debe incrementar el valor del parámetro de inicialización SHARED_POOL_SIZE. El valor máximo depende del sistema operativo.

Afinar la memoria reservada para el Shared Pool

En algunos sistemas la base de datos puede tener dificultades para encontrar espacios contiguos de memoria para satisfacer un requerimiento grande de memoria. Esta búsqueda puede corromper la Shared Pool, iniciando la fragmentación y afectando el rendimiento.

El DBA puede reservar memoria dentro de la Shared Pool para satisfacer grandes asignaciones durante operaciones como la compilación de un bloque de PL/SQL o un trigger. Los objetos pequeños no estarán fragmentados en la lista reservada, ayudando de esta manera a que la memoria tenga espacios grandes y cortos.

El tamaño de la lista reservada así como el tamaño mínimo de los objetos que están asignados en la lista reservada, están controlados por dos parámetros de inicialización: SHARED_POOL_RESERVED_SIZE y SHARED_POOL_RESERVED_MIN_ALLOC.

SHARED_POOL_RESERVED_SIZE controla la cantidad de SHARED_POOL_SIZE reservada para asignaciones grandes

SHARED_POOL_RESERVED_MIN_ALLOC controla la asignación para la memoria reservada. Para crear una lista reservada, SHARED_POOL_RESERVED_SIZE debe ser más grande que SHARED_POOL_RESERVED_MIN_ALLOC. Solamente asignaciones más grandes que SHARED_POOL_RESERVED_POOL_MIN_ALLOC pueden asignar espacio para la lista reservada.

Control del espacio de reclamación para el Shared Pool

El procedimiento ABORTED_REQUEST_THRESHOLD en el paquete DBMS_SHARED_POOL, permite a los usuario limitar el tamaño de las asignaciones permitidas para limpiar la Shared Pool si la lista reservada no puede satisfacer los requerimientos. La base de datos libera los objetos no usados de la Shared Pool mientras no haya suficiente memoria para satisfacer la asignación de requerimientos. En muchos casos, se libera bastante memoria sin encontrar un espacio grande de memoria contigua y entonces ocurre un error. Sin embargo, liberando todos los objetos se puede afectar a otros usuarios e impactar el rendimiento. El procedimiento ABORTED_REQUEST_THRESHOLD permite al DBA localizar el error para el proceso que no se puede asignar a memoria.

Valores de parámetros de inicio

Como valor inicial podemos hace que el SHARED_POOL_RESERVED_SIZE tenga un 10% del SHARED_POOL_SIZE, aunque para muchos sistemas este valor es insuficiente. El valor de dafault para SHARED_POOL_RESERVED_MIN_ALLOC es generalmente adecuado. Si se incrementa ese valor la base de datos permitiría menos asignaciones para la lista reservada y entonces se requerirá más memoria para la Shared Pool.

Idealmente se debería hacer un SHARED_POOL_RESERVED_SIZE grande para satisfacer cualquier requerimiento de memoria para la lista reservada sin liberar objetos de la Shared Pool. La cantidad de memoria del sistema operativo sin embargo puede restringir el tamaño del SGA.

En un sistema con memoria libre amplia para incrementar el SGA, la meta es tener REQUEST_MISS = 0. Si el sistema esta restringido por la memoria del sistema operativo la meta es la siguiente:

- REQUEST_FAILURES = 0 o no incrementarla
- LAST_FAILURE_SIZE > SHARED_POOL_RESERVED_MIN_ALLOC
- AVG_FREE_SIZE > SHARED_POOL_RESERVED_MIN_ALLOC

SHARED_POOL_ RESERVED_SIZE muy pequeño

La memoria Pool reservada es muy pequeña cuando:

- REQUEST_FAILURES > 0 (y se incrementa)

Y por lo menos una de las siguientes líneas es verdadera:

- LAST_FAILURE_SIZE > SHARED_POOL_RESERVED_MIN_ALLOC
- MAX_FREE_SIZE < SHARED_POOL_RESERVED_MIN_ALLOC
- FREE_MEMORY < SHARED_POOL_RESERVED_MIN_ALLOC

Se tienen dos opciones dependiendo las restricciones del SGA:

- Incrementar el SHARED_POOL_RESERVED_SIZE y la SHARED_POOL_SIZE
- Incrementar el SHARED_POOL_RESERVED_MIN_ALLOC (pero también podría ser necesario incrementar el SHARED_POOL_SIZE)

La primera opción incrementa la cantidad de memoria disponible para la lista reservada sin tener un impacto sobre los usuarios no asignados en memoria para la lista reservada. La segunda opción reduce el número de asignaciones permitidas para usar memoria de la lista reservada, sin embargo, incrementando el Shared.Pool normal puede tener un impacto sobre otros usuarios en el sistema.

SHARED_POOL_ RESERVED_SIZE muy grande

Al tener mucha memoria se podría asignar en la lista reservada. Este sería el caso si:

- REQUEST_MISS = 0 o no se incrementa
- FREE_MEMORY = > 50% del SHARED_POOL_RESERVED_SIZE mínimo

Se tienen dos opciones:

- Decrementar SHARED_POOL_RESERVED_SIZE
- decrementar SHARED_POOL_RESERVED_MIN_ALLOC (si no tiene valor de default)

SHARED_POOL_SIZE muy pequeña

La tabla V$SHARED_POOL_RESERVED puede indicar cuando la SHARED_POOL_SIZE es muy pequeña. Este puede ser el caso si:

- REQUEST_FAILURES > 0 y se incrementa
- LAST_FAILURE_SIZE < SHARED_POOL_RESERVED_MIN_ALLOC

Se tienen dos opciones, si se tiene habilitada la lista de reserva:

- decrementar SHARED_POOL_RESERVED_SIZE
- decrementar SHARED_POOL_RESERVED_MIN_ALLOC (Si tiene un tamaño grande de dafault)

En otros casos

- incrementar SHARED_POOL_SIZE

Afinar el Buffer Cache

Después de afinar las private SQL y PL/SQL áreas y el Shared Pool, se puede dedicar la afinación a la memoria restante en el buffer cache. Sería necesario repetir los pasos de asignación de memoria después del paso inicial del proceso. Los pasos subsecuentes permiten hacer ajustes en los pasos anteriores basándose en cambios en pasos subsecuentes. Por ejemplo, si aumenta el tamaño del buffer cache, se necesitaría asignar más memoria a Oracle para evitar la paginación y alternancia.

Los dispositivos físicos de entrada y salida (I/O) toman un tiempo significativo, generalmente más de 15msec, y también incrementan los recursos del CPU requeridos debido a la magnitud de los drivers y a los eventos del sistema operativo. La meta es por consiguiente reducir lo más posible el tiempo en que los bloques requeridos estén en memoria. Para oracle este término se aplica específicamente al database buffer cache.

Calculando el impacto del Cache

Oracle genera estadísticas que reflejan acceso a datos y almacena estas en una tabla dinámica llamada V$SYSSTAT.
Estas estadísticas son usadas para afinar el buffer cache:

db block gets, consistent gets La suma de los valores de estas estadísticas es la cantidad total de requerimientos para datos. Este valor incluye las satisfacciones de requerimientos a través de acceso a los buffer en memoria.
physical reads El valor para estas estadísticas es el número total de requerimientos para datos resultado de accesos a datafiles sobre el disco

Para monitorear estas estadísticas podemos usar el siguiente query:

```
SELECT name, value
FROM v$sysstat
WHERE name IN ('db block gets', 'consistent gets',  'physical reads');
```

Que da como salida la siguiente información:

```
NAME                                                              VALUE
---------------------------------------------------------------- ----------
db block gets                                                      85792
consistent gets                                                   278888
physical reads                                                     23182
```

Se puede calcular el impacto para el buffer caché con la siguiente fórmula:

Impacto = 1 - (*physical reads* / (*db block gets* + *consistent gets*))

Basado en las estadística obtenidas para el ejemplo del Query, el buffer cache tiene un impacto de 94%.

La relación entre el impacto del cache y el número de buffers esta lejos de una distribución plana. Cuando se afina el buffer Pool, se debe evitar el uso de buffers adicionales lo cual contribuye a que sea pequeño el impacto del cache o que no exista. Como se ilustra en la figura 22, hay solo lazos estrechos entre los valores de DB_BLOCK_BUFFERS que son considerados. El efecto es completamente intuitivo

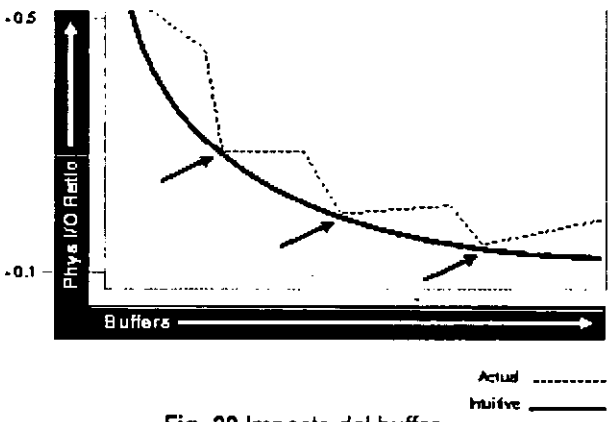


Fig. 22 Impacto del buffer.

Un error común es continuar incrementado el valor del DB_BLOCK_BUFFERS cuando el último incremento hecho no generó ninguna diferencia. Esto es debido a que operaciones como un full table scan no usan el buffer pool

Como reglas para incrementar el DB_BLOCK_BUFFERS tenemos:

- El impacto del cache es menor que 0.9
- Los incrementos previos del DB_BLOCK_BUFFERS fueron efectivos

Afinación de las Sort Areas

Si ocurren comúnmente grandes ordenamientos se debe considerar el incrementar el valor del parámetro SORT_AREA_SIZE teniendo estas dos metas en mente:

- Aumentar la cantidad de ordenamientos que pueden hacerce por completo en memoria
- Aumentar la velocidad de los ordenamientos que no se puedan generar enteramente en memoria.

Se pueden usar áreas de ordenamiento grandes efectivamente si se combina un SORT_AREA_SIZE grande con un SORT_AREA_RETAINED_SIZE pequeño. Si la memoria no es liberada cuando se desconectan los usuarios de la base de datos, las áreas grandes de ordenamiento pueden causar problemas. El parámetro SORT_AREA_RETAINED_SIZE especifica al DBA que la memoria debe ser liberada tan pronto como sea posible. Se debe fijar el parámetro a cero si las áreas de ordenamiento grande serán usadas en el sistema por muchos usuarios simultáneamente

Reasignación de memoria

Después de reasignar la memoria de las estructuras de oracle, se debe reevaluar el rendimiento del library cache, el data dictionary cache y el buffer cache. Si se redujo el consumo de memoria por alguna de esas estructuras se puede necesitar asignar más memoria a otras estructuras. Por ejemplo, si se ha reducido el tamaño del buffer cache se puede necesitar agregar más memoria para su uso por el library cache.

Mientras se reasigna la memoria se puede determinar que el tamaño óptimo para las estructuras de memoria de oracle requieren más memoria de la que el sistema operativo provee, en este caso se puede mejorar el rendimiento adicionando más memoria a la computadora.

Reducción del total de memoria usada

Si el problema de rendimiento es simplemente por que el servidor no tiene suficiente memoria para correr las aplicaciones como esta configurado actualmente y lógicamente la aplicación es una sola, entonces hay solamente dos posibles soluciones:

- Incrementar la cantidad de memoria disponible
- Decrementar la cantidad de memoria usada

Las reducciones más dramáticas de uso de memoria del servidor serán siempre de reducir el número de conexiones a la base de datos, esto puede resolver el uso de conexiones abiertas en la red y el número de procesos del sistema operativo, sin embargo, el reducir el número de conexiones sin reducir la cantidad de usuarios ocasiona que las conexiones que quedan se tienen que compartir. Esto forza al usuario a que cada requerimiento a la base de datos se convierta en una transacción completa.

La arquitectura multitarea de oracle (MTS) representa una solución que es altamente efectiva al reducir la cantidad de procesos del sistema operativo sobre el servidor pero menos efectivo en reducir el requerimiento de memoria global. El uso de MTS no tiene efecto sobre la cantidad de conexionas a la red.

8. AFINACIÓN DE LOS DISPOSITIVOS DE ENTRADA Y SALIDA (I/O)

El rendimiento de muchos software de aplicación está inherentemente limitado al I/O del disco. Comúnmente la actividad del CPU puede ser suspendida mientras se completa la actividad del I/O. Oracle está diseñado para que el rendimiento no esté limitado por el I/O. Al afinar el I/O se puede ayudar a mejorar el rendimiento cuando el disco pueda operar en forma ilimitada, sin embargo, el afinar el I/O no ayuda cuando el CPU esta limitado.

Cuando se diseña un sistema nuevo se debe analizar las necesidades de I/O determinando que recursos se requerirán para alcanzar el rendimiento deseado. Para un sistema existente se debe hacer la afinación del I/O tomando en cuenta las siguientes consideraciones:

- Determinar la cantidad de discos en el sistema
- Determinar la cantidad de discos que están siendo utilizados por oracle
- Determinar el tipo de I/O que el sistema ejecuta
- Determinar como asignar objetos sobre múltiples discos.
- Calcular en nivel de rendimiento que se espera.

Para determinar si una aplicación correrá mejor en los discos que tienen disponibles, si se almacenan los datos en dispositivos o directamente en file systems se debe seguir el diseño de almacenamiento de archivos siguiente:

- Identificar las operaciones requeridas por la aplicación
- Probar el rendimiento de los discos del sistema a través de diferentes operaciones requeridas por la aplicación
- Evaluar qué tipo de discos se tienen para un mejor rendimiento para operaciones que predominen en una aplicación

Evaluar la aplicación para determinar que tan a menudo esta requiere cada tipo de operación de I/O. La siguiente tabla muestra los tipos de operaciones de lectura y escritura ejecutadas por cada proceso background y foreground

Operation Process	*LGWR*	*DBWR*	*ARCH*	*SMON*	*PMON*	*CKPT*	*Fore-ground*
Lectura Secuencial			X	X		X	X
Escritura Secuencial	X		X			X	
Lectura Aleatoria				X			X
Escritura Aleatoria		X					

Conociendo los tipos de operaciones que predominan en una aplicación y la velocidad con la cual un sistema en particular puede procesar los correspondientes I/Os se puede elegir el disco que maximize el rendimiento del sistema. El número de I/Os de un disco puede depender de cuantas operaciones de lectura y escritura relacionadas a objetos almacenados en dispositivos o en archivos, esto afecta la cantidad de discos que se usan para alcanzar el nivel de rendimiento deseado.

Detección de problemas de I/O

Cuando se sospecha que existen problemas de I/O se pueden evaluar dos áreas

- Checar la utilización del I/O del sistema
- Checar la utilización del I/O de oracle

Oracle genera estadísticas que reflejan el acceso al disco para los database files, sin embargo estas estadísticas solo reportan La utilización del I/O de las sesiones de oracle.

Utilización del I/O del sistema

Se pueden usar herramientas del sistema operativo para monitorear que procesos corren sobre el sistema en conjunto y para monitorear el acceso a disco de todos los archivos. Los discos mantienen datafiles y redo log files y archivos no relacionados con oracle, se puede intentar reducir algunos acceso pesados a los discos que contengan database files.

Utilización del I/O por Oracle

La siguiente tabla muestra en cuales tablas se pueden checar estadísticas de I/O relacionadas con database files, redo log files, archive files y control files de oracle

Tipo de archivo	Dónde se encuentran las estadísticas
Database Files	V$FILESTAT
Log Files	V$SYSSTAT, V$SYSTEM_EVENT, V$SESSION_EVENT
Archive Files	V$SYSTEM_EVENT, V$SESSION_EVENT
Control Files	V$SYSTEM_EVENT, V$SESSION_EVENT

La siguiente tabla lista los procesos que reflejan estadísticas de I/O para tipos diferentes de archivos oracle

File Process	LGWR	DBWR	ARCH	SMON	PMON	CKPT	Fore-ground
Database Files		X		X	X	X	X
Log Files	X						
Archive Files			X				
Control Files	X	X	X	X	X	X	X

Si existen problemas relacionados con I/O de oracle hay que afinar estos, pero si los procesos no consumen los recursos de I/O disponibles, entonces son otros procesos y hay que regresar al sistema e identificar estos.

Se puede examinar el acceso a los datafiles del disco por medio de la tabla V$FILESTAT, esta muestra los siguientes ítems:

- Cantidad de lecturas y escrituras físicas
- Cantidad de bloques leídos y escritos
- Total de tiempo de I/O entre lecturas y escrituras

Las siguientes columnas reflejan la cantidad de accesos al disco para cada datafile:

PHYRDS El valor de esta columna es la cantidad de lecturas para cada database file.
PHYWRTS El valor de esta columna es la cantidad de escrituras para cada database file.

Se puede usar el siguiente query para monitorear esos valores sobre un periodo en el tiempo mientras la aplicación está corriendo

```
SELECT name, phyrds, phywrts
FROM v$datafile df, v$filestat fs
WHERE df.file# = fs.file#;
```

Este query recupera también el nombre de cada datafile y su salida es la siguiente:

```
NAME                                       PHYRDS     PHYWRTS
---------------------------------------- ---------- ----------
/oracle/ora70/dbs/ora_system.dbf             7679       2735
/oracle/ora70/dbs/ora_temp.dbf                 32        546
```

El total de I/O para un disco es la suma de las columnas PHYRDS y PHYWRTS para todos los database files manejados por una instancia oracle en el disco. Se debe determinar el valor de cada uno de los discos y determinar el impacto de ocurrencia de I/O para cada disco.

Solución de problemas de I/O

Para solucionar problemas de I/O se deben considerar los siguientes puntos:

- Reducir la contención a través de la distribución de I/O
- Manejar el espacio del disco dinámicamente
- Afinar ordenamientos
- Afinar Checkpoints
- Afinar LGWR y DBWR I/O

9. AFINACIÓN DE CONTENCIÓN DE RECURSOS

Los síntomas de problemas de contención de recursos pueden ser encontrados en la vista V$SYSTEM_EVENT. Esta vista revela varios síntomas que pueden impactar el rendimiento tales como latch contention, buffer contention, I/O contention. Es importante recordar que estos son solamente síntomas y no las causas actuales.

Por ejemplo, si se observa V$SYSTEM_EVENT se nota que el buffer tiene mucho tiempo de espera. Si se diera el caso de que muchos procesos se insertan en el mismo bloque, pero deben esperar cada uno antes de poder hace una inserción se dice que hay contención. La solución podría ser introducir una free list para el objeto en cuestión. El buffer ocupado puede causar también algún tiempo de espera para las free list, la mayor parte de estas esperas es causada por misses en el buffer cache, este es también un efecto de intentar insertar dentro del mismo bloque. En lugar de incrementar el SPINCOUNT se debería cambiar el objeto y asignarlo a múltiples proceso de inserción en bloques libres, esto generaría una reducción en la contención

Los problemas más comunes de contención son los siguientes:

Contención para Segmentos de Rollback
Contención para procesos de Servidores Multitarea
Contención para Redo Log Buffer Latches

Reducción de contención para Segmentos de Rollback

La contención en los segmentos de Rollback es reflejada por contención en los buffers que contienen los bloques de segmentos de Rollback. Se puede determinar si la contención de segmentos de Rollback esta reduciendo el rendimiento checando la tabla V$WAITSTAT. Esta tabla contiene estadísticas que reflejan la contención de bloques, las estadísticas reflejan la contención de diferentes clases de bloques:

system undo header	Es la cantidad de espera de los buffers contenida en la cabecera de los bloques del segmento de Rollback SYSTEM
system undo block	La cantidad de espera de los buffers contenidos en cabeceras de otros bloques del segmento de Rollback SYSTEM
undo header	La cantidad de espera de buffers contenidos en la cabecera de otros segmentos de Rollback diferentes a SYSTEM
undo block	La cantidad de espera para buffers que contienen bloques de segmentos de Rollback diferentes a SYSTEM

Se puede usar el siguiente query para monitorear estadísticas sobre un periodo de tiempo mientras la aplicación está corriendo

```
SELECT class, count
FROM   v$waitstat
WHERE class IN ('system undo header', 'system undo block',
      'undo header', 'undo block');
```

El resultado de este query arroja la siguiente salida:

```
CLASS               COUNT
------------------ ----------
system undo header       2089
system undo block         633
undo header              1235
undo block                942
```

Comparando la cantidad de espera para cada clase de bloque con la cantidad total de requerimientos de datos para un mismo periodo de tiempo, se puede monitorear la cantidad total de requerimientos de datos para un periodo de tiempo con este query:

```
SELECT SUM(value)
FROM   v$sysstat
WHERE  name IN ('db block gets', 'consistent gets');
```

El query genera la siguiente salida:

```
SUM(VALUE)
----------
    929530
```

si la cantidad de espera para algunas clases es mayor que el 1% de la cantidad total de requerimientos, se debe considerar crear más segmentos de Rollback para reducir la contención.

Contención de procesos Dispatcher

La contención de procesos dispatcher puede ser reflejo de estos sintomas;
cadencia

alto grado de ocupación de los procesos dispatcher existentes
mantenimiento del incremento de tiempo de respuesta de procesos dispatcher existentes

La tabla V$DISPATCHER contiene estadísticas que reflejan la actividad de los procesos dispatcher. Estas columnas reflejan la actividad de grado de ocupación de los procesos dispatcher:

IDLE Esta columna es el tiempo de inactividad de los procesos dispatcher en centésimas de segundo
BUSY Esta columna muestra el tiempo de ocupación de los procesos dispatcher en centésimas de segundo

Se puede usar el siguiente query para monitorear estas estadisticas en un periodo de tiempo cuando una aplicación esta corriendo.

```
SELECT network "Protocol",
       SUM(busy)/(SUM(busy)+SUM(idle))"Total Busy Rate"
FROM v$dispatcher
GROUP BY network;
```

Este query retorna el total de tiempo de ocupación del proceso dispatcher para cada protocolo; esto es, el porcentaje de tiempo que cada proceso dispatcher ocupa para cada protocolo. El resultado del query es el siguiente:

```
Protocol  Total Busy Rate
--------  ---------------
decnet          .004589828
tcp             .029111042
```

Para este resultado se pueden hacer las siguientes observaciones:
DECnet del proceso dispatcher está ocupado casi un 0.5% del tiempo.
TCP del proceso dispatcher está ocupado casi un 3% del tiempo.

Si la base de datos se usa solamente 8 horas por dia, las estadisticas necesitan ser normalizadas para tiempos efectivos de trabajo, no se deben solamente observar las estadísticas en el momento en que la instancia se inició, más bien, se deben verificar las estadísticas en las horas de mayor carga de trabajo, así el proceso dispatcher par un protocolo específico estará ocupado más de un 50% del tiempo efectivo de carga de trabajo. Así podremos saber que se necesitan agregar procesos dispatcher que pueden mejorar el rendimiento de usuarios conectados a oracle usando ese protocolo.

Para agregar procesos dispatcher mientras oracle está corriendo, se debe usar el parámetro MTS_DISPATCHERS del comando ALTER SYSTEM. La cantidad total de procesos dispatcher para todos los protocolos está limitado por el valor del parámetro de inicialización MTS_MAX_DISPATCHERS. Se puede necesitar incrementar ese valor antes de agregar procesos dispatcher, el valor de default para este parámetro es 5 y el valor máximo depende del sistema operativo

Contención de Shared Server Processes

La contención de shared server processes puede ser reflejada por el incremento en el tiempo de espera para los requerimientos. La tabla V$QUEUE contiene estadísticas que reflejan el tiempo de espera para un shared server processes.

WAIT Esta columna muestra el tiempo total de espera en centésimas de segundo para requerimientos que se han hecho.
TOTALQ Esta columna muestra la cantidad total que el requerimiento ha estado en cola de espera

Se puede usar el siguiente query para monitorear las estadísticas mientras una aplicación está corriendo

```
SELECT DECODE( totalq, 0, 'No Requests',
       wait/totalq || ' hundredths of seconds')
       "Average Wait Time Per Requests"
FROM   v$queue
WHERE  type = 'COMMON';
```

Este query retorna el tiempo total de espera de todos los requerimientos y la cantidad total de estos requerimientos en cola de espera. El resultado del query es el siguiente:

```
Average Wait Time per Request
-----------------------------
.090909 hundredths of seconds
```

Para este resultado se puede decir que el tiempo de espera es en promedio de 0.09 centésimas de segundo en la cola de espera antes de ser procesados.

Oracle automáticamente agrega shared server processes si la carga se incrementa dramáticamente, es poco probable mejorar el rendimiento agregando simplemente más shared server processes, sin embargo, la cantidad de shared server processes se ha hecho más grande por medio del parámetro de inicialización MTS_MAX_SERVERS y el promedio del tiempo de espera en los requerimientos en cola aun están creciendo, se puede entonces mejorar el rendimiento incrementando el valor de MTS_MAX_SERVERS. El valor de default para este parámetro es 20 y el valor máximo depende del sistema operativo.

Contención de los Redo Log Buffer

La contención de los redo log buffer se incrementa rara vez inhibiendo el rendimiento de la base de datos, sin embargo oracle provee métodos para monitorear y reducir la contención de los redo log buffer cuando esta ocurre.

Cuando el LGWR escribe entradas redo desde el redo log buffer al redo log file, el proceso puede copiar nuevas entradas sobre las que ya han sido escritas en el disco. LGWR normalmente escribe bastante rápido para asegurar que el espacio siempre esté disponible para las nuevas entradas en el buffer, de la misma forma cuando el acceso al redo log es pesado.

Las estadísticas *redo log space requests* reflejan la cantidad de tiempo que un proceso de usuario espera para espacio en el redo log buffer. Estas estadísticas están disponibles en la tabla V$SYSSTAT. Se puede usar el siguiente comando para monitorear estas estadísticas:

```
SELECT name, value
FROM v$sysstat
WHERE name = 'redo log space requests';
```

El valor del *redo log space requests* debería estar cerca de cero. Si este valor se incrementa constantemente, el proceso tiene que esperar por espacio en el buffer. Esto puede causar que el log buffer comience a ser pequeño. Se debe entonces incrementar el tamaño del redo log buffer si es necesario cambiando el parámetro de inicialización LOG_BUFFER. El valor de este parámetro expresado en bytes debe ser múltiplo del DB_BLOCK_SIZE. Alternativamente se deben mejorar los checkpoint o los proceso de archivación.

Contención de Redo Log Buffer Latches

El acceso a los redo log buffers es generalmente regulado por latches. Existen dos tipos de control latch para accesar el redo log buffer: el redo allocation latch y el redo copy latches

El Redo Allocation Latch

El *redo allocation latch* controla la asignación de espacio para entradas redo en el redo log buffer. Para asignar espacio en el buffer, un proceso de usuario de oracle debe obtener una redo allocation latch. Para una sola redo allocation latch solamente un proceso de usuario puede asignar espacio en el buffer al mismo tiempo. La redo allocation latch forza a una secuencia natural de entradas en el buffer. Después de asignar espacio para una entrada redo, el proceso de usuario podría copiar la entrada en el buffer. Un proceso puede solamente copiar sobre la redo allocation latch si la entrada redo es pequeña.

El tamaño máximo para una entrada redo que puede ser copiada en la redo allocation latch está especificado en el parámetro de inicialización LOG_SMALL_ENTRY_MAX_SIZE. El valor de este parámetro está expresado en bytes. El valor mínimo, máximo y el de default dependerán del sistema operativo.

Redo Copy Latches

El copy latch es obtenido primero, después el allocation latch, la asignación es ejecutada y liberada. Después la copia es ejecutada bajo en copy latch y el copy latch es liberado. El allocation latch se mantiene así por un breve periodo de tiempo y el sistema no intenta obtener el copy latch mientras se mantenga el allocation latch

Si la entrada redo es grande para ser copiada sobre el redo allocation latch, el proceso de usuario debe obtener un *redo copy latch* antes de copiar la entrada en el buffer. Mientras se mantenga un redo copy latch el proceso de usuario copiará la entrada redo en un espacio asignado en el buffer y se liberará el redo copy latch.

Si la computadora tiene múltiples CPUs, el redo log buffer puede tener múltiples redo copy latches, esto permitiría que múltiples procesos pudieran copiar entradas en el redo log buffer concurrentemente. La cantidad de redo copy latches está determinada por el parámetro de inicialización LOG_SIMULTANEOUS_COPIES. El valor de default para este parámetro es el número de CPUs disponibles en la instancia oracle.

Una computadora con un solo CPU podría no tener redo copy latches, solo un proceso podría ser activado una sola vez, todas las entradas redo serían copiadas sobre el redo allocation latch siendo indiferente su tamaño.

Accesos pesados al redo log buffer pueden resultar en contención para los redo log buffer latches. Esto puede reducir el rendimiento. Oracle genera estadísticas para la actividad de todos los latches y se almacenan en la tabla V$LATCH.

Cada registro en la tabla V$LATCH contiene estadísticas para diferentes tipos de latch. Las columnas de la tabla reflejan la actividad para diferentes tipos de requerimientos de latch.

Existen más casos re contención de latch cuando dos o más procesos al mismo tiempo intentan obtener el mismo latch. La contención de latch ocurre rara vez en computadoras que tienen un solo CPU donde solamente un solo proceso puede ser activado una sola vez.

Para reducir la contención del redo allocation latch se debe minimizar el tiempo que un solo proceso mantenga al latch. Para reducir el tiempo, se debe reducir la copia sobre el redo allocation latch. Decrementando el valor del parámetro de inicialización LOG_SMALL_ENTRY_MAX_SIZE se reduce la cantidad y el tamaño de las entradas redo copiadas sobre el redo allocation latch.

Para computadoras con múltiples CPUs, múltiples redo copy latches permiten múltiples proceso para copiar entradas para el redo log buffer concurrentemente. El valor de default de LOG_SIMULTANEOUS_COPIES es el número de CPUs disponibles para la instancia de oracle

Si se observa contención para el redo copy latches, hay que agregar más latches para incrementar la cantidad de redo copy latches. Esto puede ayudar a tener hasta dos veces el valor de redo copy latches.

Para concluir podemos comentar que la técnicao el método más adecuado dependen generalmente de las dimensiones de la base de datos así como del sistema operativo, por eso es importante conocer todas e incluso llevarlas a nivel de detalle para garantizar una base de datos funcional que es el objetivo de un administrador de base de datos Oracle.

CONCLUSION

Hemos llegado pues al punto final de esta investigación y puedo comentar con satisfacción que los objetivos se han cumplido, el cometido principal que me llevo a realizar este trabajo fue el determinar las tareas del oracle dba con respecto a la seguridad de la base de datos, el respaldo y recuperación de los datos y la afinación de sus procesos a fin de asimilar el conjunto de conceptos que se deriven de esas actividades y poner en marcha un plan en el que se distribuyan estas tareas en forma eficiente logrando con ello la estabilidad en los sistemas de información a los que da servicio el oracle server y satisfacción de los usuarios.

Con el desarrollo de los capítulos se cumplió en forma amplia y concisa la meta fijada, ya que una vez asimilados los conceptos de la base de datos oracle entonces podemos enfrentar esta tarea en forma eficiente, no importando la plataforma o sistema operativo sobre la que se desea ser montado el oracle server, debido a que la arquitectura está estructurada de la misma forma y lo que cambia son ligeras instrucciones entre estas.

Ahora se cuenta con un conjunto de elementos que nos permitirán hacer propuestas para eficientar los recursos con que actualmente cuente cualquier empresa que use el oracle server, así como poder proponer nuevas características en diseño de la base de datos en sus niveles lógicos y físicos. Con acceso a la información que aquí se presente, una persona que ocupe una gerencia o participe en la toma de decisiones puede de una manera sencilla verificar que tareas son más complejas y cuales son más sencillas y dar así precedencia a los problemas derivados de la administración del oracle server por ejemplo.

Además como he comentado con anterioridad, las pocas referencias bibliográficas en español, ocasionan que personas que estén limitadas en el sentido de que entiendan poco del idioma inglés, tengan un punto de apoyo en este trabajo y debido a la estructura lógica en como está organizado, es de un sencillo entendimiento. La única observación con respecto al contenido es que está dirigido a una personas que tengan nociones de bases de datos, sistemas de información, sistemas operativos y redes de computadoras, ya que constantemente se hace referencia a alguno de estos rubros.

Una vez más aprovecho para comentar que muchos de los conceptos los manejé sin traducción pues de hecho es poco común e incluso raro que se hagan referencias a estos traducidos, inclusive, cuando se toman cursos o se participa en alguna junta donde se traten temas relacionados al oracle server siempre se manejan de la manera que aquí utilice.

BIBLIOGRAFÍA

AMSTRONG, Eric, Oracle7 Server SQL Languaje Quick Reference, Oracle Corporation, U.S.A., 1992. PP . 12-30.

BOBROWSKI, Steven, Oracle 7 Server Concepts, U.S.A., Oracle Corporation, Oracle Technology Network, 1996. URL : http://technet.oracle.com/docs/products/oracle7/doc_index.htm, Cap. 1,2,3,4,5,6,7 y 9

FRANZZINI, John, Oracle 7 Server Tuning, U.S.A., Oracle Corporation, Oracle Technology Network, 1996. URL : http://technet.oracle.com/docs/products/oracle7/doc_index.htm. Cap. 1,2,3,7,8,10,12,13,14y 16

GRIFFIN, Don, et. al., V7.2 Tuning Aplications,U.S.A. Oracle Corporation, 1994. Cap 3, 4 y 5

GURRY, Mark, Oracle Performance Tuning ,U.S.A, O'Reilly, Segunda Edición 1998. Cap. 1,2,6,7 y 10.

LONEY, Kevin, Oracle DBA Handbook,U.S.A. Oracles Press, Osborne McGraw-Hill,1998. Cap 1,2 y 3

SANAWALLA, Noorali, Advanced Oracle Tunning and Database Administración, U.S.A., Oracles Press, Osborne McGraw-Hill,1997, Cap. 1, 2,5,7,9,10 y 11.

SHEMELKES, Corina, Manual para la presentación de anteproyectos e informes de investigación (Tesis), 1er ed., México, Ed Harla,1994, Cap. 1-4